PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-215394
(43)Date of publication of application : 02.08.2002

(51)Int.Cl. G06F 9/44

(21)Application number : 2001-241749
(22)Date of filing : 09.08.2001

(71)Applicant : FUJITSU LTD
(72)Inventor : MATSUZUKA TAKAHIDE
NOMURA YOSHIHIDE

(30)Priority

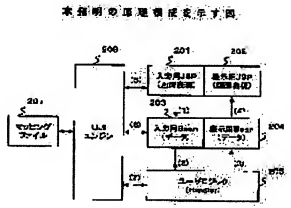
Priority number : 2000246139 Priority date : 15.08.2000 Priority country : JP
2000347977 15.11.2000 JP

(54) WEB APPLICATION DEVELOPMENT AND EXECUTION SYSTEM AND WEB APPLICATION GENERATING DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a framework described by dividing into modules of data, logic, and screen, when an application server for executing a Web application is developed.

SOLUTION: This Web application development and execution system comprises an input content transformation means 206 for transforming the input content 201 of an Web page into a data object 203, a first external definition file 207 for mapping the combination of the type of the data object 203 with the command thereof an each processing routine, a processing logic 205 having a plurality of processing routines, a processing routine determination means 206 for determining an appropriate processing routine from the processing logic 205, based on the type and command of the data object 203 and the external definition file 207, and a second external definition file 207 for mapping the combination of the results of the processed result of the processing logic 205 with the type of the data object 204 on a display component 202.



LEGAL STATUS

[Date of request for examination]	05.08.2004
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] The Web system characterized by being the Web system which returns a response to a client, changing said request into a data object, processing [to process the request from a client by the server,] it by said server, and changing and returning the data object which it is as a result of processing to the response to said client.
[Claim 2] A contents conversion means of an input to be a system for developing and performing Web application, and to change the contents of an input into a data object, Processing logic equipped with two or more manipulation routines, and the class of said data object and the 1st external declaration file which maps the combination of a command in said each manipulation routine, The Web application development and the executive system characterized by having a manipulation-routine decision means to determine a suitable manipulation routine from the manipulation routine with which said processing logic is equipped based on the class of said data object, said command, and said 1st external declaration file.
[Claim 3] They are the Web application development and the executive system which are Web application development and an executive system according to claim 2, and is characterized by for said contents conversion means of an input make the specific item of the data input page described in HTML the class name of said data object, make the identifier of each input column prepared in this page for data inputs correspond to the attribute of said data object, and generate the program for data objects automatically.
[Claim 4] The Web application development and the executive system characterized by having the processing logic which is a system for developing and performing Web application, and is equipped with two or more manipulation routines, and the processing result of said processing logic and the 2nd external declaration file which maps the combination of the class of data object in the component for a display.
[Claim 5] The Web application development and the executive system characterized by having the template file which specified the method of arrangement of two or more components for a display which are Web application development and an executive system according to claim 4, and are further arranged to the page to display, and outputting the processing result of two or more of said processing logic based on said template file.
[Claim 6] The XML mapping file which are Web application development and an executive system according to claim 2, and maps the tag and data object of XML further, XML which performs the interconversion of the tag of XML, and a data object Data It has a Binding engine. It is said XML when the tag described by XML as said contents of an input is received. Data A Binding engine Said XML which received is changed into said data object based on the tag and said XML mapping file of said XML which received. Said manipulation-routine decision means Based on said data object, the tag of said XML which received, and said 1st external declaration file, a suitable manipulation routine is determined from the manipulation routine in said processing logic. Said XML Data A Binding engine is the Web application development and an executive system characterized by what the data object obtained as a processing result of said processing logic is changed and outputted for to the tag of XML.
[Claim 7] They are the Web application development and the executive system which are Web application development and an executive system according to claim 6, and is characterized by said

XML mapping file mapping the tag of XML which performs the same processing as the data based on a certain HTTP in the data object of the same class as the data based on said a certain HTTP.
[Claim 8] The step which is the record medium which recorded the program read by it when used by computer, and changes the contents of an input into a data object, The external declaration file which maps the combination of the class of said data object, a command, and the class of said data object and a command in each manipulation routine, The record medium which recorded the program for making the step which is alike, is based and determines a suitable manipulation routine from the manipulation routine in processing logic perform to said computer and which can be computer read.
[Claim 9] Each object which processes the request from a client by the server, is the Web system which returns a response to a client, and is processed according to said request by said server The attribute definition about a time scope based on the relative physical relationship on a time-axis in case each is processed is carried out. Said server The Web system characterized by branching to the object by which the small time scope is defined, and advancing processing of the object according to said request from the object by which the bigger time scope is defined.
[Claim 10] The Web system characterized by performing the object which the request from a client is processed by the server, it is [object] the Web system which returns a response to a client, and actuation of processing by this manipulation routine is supervised [object] at the time of the manipulation-routine call by the server, and makes advance of this processing continue by this server.
[Claim 11] It is prepared in the server side which delivers and receives various kinds of data between clients. It is Web application generation equipment which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed. A data object storing means by which the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data is stored, A definition statement storing means by which the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML is stored, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement Web application generation equipment characterized by having a HTML sentence generation means to generate the HTML sentence expressing the Web page screen where these data are displayed.
[Claim 12] It is Web application generation equipment which has further a data class acquisition means to acquire the data class which is Web application generation equipment according to claim 11, and is the attribute defined corresponding to said DS, and is characterized by what said HTML sentence generation means chooses for said definition statement related with the data which said data object has based on said data class.
[Claim 13] It is Web application generation equipment according to claim 11. A request acquisition means to acquire this request that is a request containing the input data to the entry form shown in the Web page screen expressed by the HTML sentence generated by said HTML sentence generation means, and is sent from said client, It is the character string contained in the HTML sentence which is contained in said request with said data, and which was generated by said HTML sentence generation means. It is based on this character string that consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface used as the foundation of generation of this HTML sentence, and this interface. Web application generation equipment characterized by having further a renewal means of data to update the data of the specific location in the DS expressed by the interface specified by this character string to the data contained in this request.
[Claim 14] The character string which is Web application generation equipment according to claim 13, and is contained in said request with said data To the character string which consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface used as the foundation of generation of said HTML sentence, and this interface Furthermore, it changes combining the character string which consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface showing the DS which the data constellation of this location has, and this interface. Said renewal means of data is Web application generation equipment characterized by what the data of the specific location in the DS which it has further in the specific location in the DS specified by said

character string are updated for to the data contained in said request.
[Claim 15] It is the approach of generating the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed by the server side which delivers and receives various kinds of data between clients. The data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data is acquired. The definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML is acquired. By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The Web application generation method characterized by having ** which generates the HTML sentence expressing the Web page screen where these data are displayed.
[Claim 16] By the server side which delivers and receives various kinds of data between clients, by performing a computer It is the record medium which recorded the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed perform to this computer. The control which acquires the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data, The control which acquires the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The storage which memorized the control program which makes the control which generates the HTML sentence expressing the Web page screen where these data are displayed perform to a computer.
[Claim 17] It is prepared in the server side which delivers and receives various kinds of data between clients. A processing logic storing means by which the processing logic by which it is Web application generation equipment which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed, and the contents of processing are defined is stored, An execution condition storing means by which the execution condition of said processing logic is stored, and a processing logic name generation means to generate the character string used as the name of said processing logic, It is the HTML sentence which calls said processing logic using said character string. Web application generation equipment characterized by having a HTML sentence generation means to generate this HTML sentence that makes processing in which this processing logic is called and performed when the event corresponding to said execution condition occurs in said client perform to this client.
[Claim 18] It is Web application generation equipment according to claim 17, and said two or more processing logic is stored in said processing logic storing means. When either of said each execution condition of two or more of said processing logic is the same It has further a processing logic generation means by which this execution condition generates the processing logic which performs this same processing logic in order. Said character string generation means The character string used as a respectively different name to said two or more processing logic and the processing logic generated by said processing logic generation means is generated. Said HTML sentence generation means It is the HTML sentence which calls the processing logic generated by said processing logic generation means using said character string. Web application generation equipment characterized by what this HTML sentence that makes processing in which this processing logic is called and performed when the event corresponding to said same execution condition occurs in said client perform to this client is generated for.
[Claim 19] It is the Web application generation method which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed by the server side which delivers and receives various kinds of data between clients. The character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined is generated. It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The Web application generation method characterized by what this HTML sentence that makes the processing to say perform to this client is generated for.
[Claim 20] By the server side which delivers and receives various kinds of data between clients, by

performing a computer It is the record medium which recorded the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed perform to this computer. The control which generates the character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined, It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The storage which memorized the control program which makes the control which generates this HTML sentence that makes the processing to say perform to this client perform to a computer.

[Claim 21] The program for performing the step which determines a suitable manipulation routine as a computer from the manipulation routine in processing logic based on the external declaration file which maps the combination of the step which changes the contents of an input into a data object, the class of said data object, a command, and the class of said data object and a command in each manipulation routine.

[Claim 22] By the server side which delivers and receives various kinds of data between clients, by performing a computer It is the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed perform to this computer. The control which acquires the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data, The control which acquires the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The control program for making the control which generates the HTML sentence expressing the Web page screen where these data are displayed perform to a computer.

[Claim 23] By the server side which delivers and receives various kinds of data between clients, by performing a computer It is the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed perform to this computer. The control which generates the character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined, It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The control program for making the control which generates this HTML sentence that makes the processing to say perform to this client perform to a computer.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the technique which raises the development effectiveness and the maintainability of the business application software which processes the basic business of a company etc.
[0002]
[Description of the Prior Art] In business applications, such as a data entry system in company business, and a workflow system, as shown in drawing 47 , the method of operating the data which a database server 5402 manages through an application server 5401 from a client 5403 has been taken. In recent years, such a business application came to be realized as Web application.
[0003] And more various attempts than before have been made by development of the application server 5401 which plays the role which controls a web browser. There is an application server 5401 using Servlet (servlet) and JSP (Java(trademark) Server Pages) as a typical thing.
[0004] As shown in drawing 48 , the application server 5401 using Servlet describes HTML (Hyper Text Markup Language)5501 which specifies a screen display, the screen data 5502, and logic 5503 (handling of an input of a screen, the check of input data, processing of data, transfer of the data to a database server 5402, etc. are performed) as one module, and is realized.
[0005] moreover, the application server 5401 using JSP is shown in drawing 49 -- as -- Java the module which describes as a module which consists of the screen data 5602 and HTML5601 which were stored in an object called Bean (Java and Java Beans is the trademark of Sun Microsystems, Incorporated), and logic 5603, or consists of HTML5601', and screen data 5602' and logic 5603 -- ' -- Java It describes and realizes so that it may store in a Bean object.
[0006] Generally, by the business application, since the data to treat become abundant, an application server 5401 divides data (screen data), logic, and screens (HTML etc.) into each module, describes them, is realized, and has a demand of wanting to reuse each module. However, as shown in drawing 48 or drawing 49 , in the application server 5401 using the conventional techniques, such as Servlet and JSP, separation of these modules was difficult.
[0007]
[Problem(s) to be Solved by the Invention] Then, in case the technical problem of this invention develops the application server which performs Web application, it is by offering data, logic, and the framework that divides into each module of a screen and is described to improve the development effectiveness and the maintainability of Web application.
[0008]
[Means for Solving the Problem] In order to solve the technical problem mentioned above, in this invention, Web application is divided into HTML with a custom-made tag, a data object, and logic, between HTML with a custom-made tag and logic is mapped by the mapping file, and the data object was used for the exchange of the data between both.
[0009] A contents conversion means of an input to change the contents of an input of a Web page into a data object if this invention is caused like 1 voice, Processing logic equipped with two or more manipulation routines, and the class of said data object and the 1st external declaration file which maps the combination of a command in said each manipulation routine, A manipulation-routine decision means to determine a suitable manipulation routine from the manipulation routine with

which said processing logic is equipped based on the class of said data object, a command, and said 1st external declaration file, It has the processing result of said processing logic, and the 2nd external declaration file which maps the combination of the class of data object in the component for a display.

[0010] Moreover, a means to acquire the data object which mounts the interface which has another data of this invention with which a client will be provided if it depends like 1 voice, and expresses the DS of these data, A means to acquire the definition statement which defines the method of presentation in the Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement, it has a means to generate the HTML sentence expressing the Web page screen where these data are displayed.

[0011] Moreover, a means to generate the character string used as the name of this processing logic that is further another processing logic of this invention by which the contents of processing are defined if it depends like 1 voice, and is prepared beforehand, It is the HTML sentence which calls said processing logic using said character string. It has a means to generate this HTML sentence that makes processing in which this processing logic is called and performed when the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in a client perform to this client.

[0012] By taking such a configuration, cooperation of the logic which specifies HTML which specifies a screen display in Web application system, a data object, and the contents of processing will serve as a non-dense, respectively, the reusability of each module can improve, and development effectiveness and maintainability can be raised.

[0013]

[Embodiment of the Invention] Hereafter, the gestalt of operation of this invention is explained, referring to a drawing.

[0014] Drawing 1 is drawing showing the outline of this invention. this invention is carried out -- UJI (Uji) -- an application server consists of HTML101 which specifies a screen display, screen data 102, and three modules of logic 103 so that it may illustrate. In addition, if HTML101 can specify a screen display, which thing is sufficient as it.

[0015] Drawing 2 is drawing showing the principle configuration of this invention. The thing corresponding to [the things corresponding to HTML101 of drawing 1 are JSP201 for an input and JSP202 for a display, and] the screen data 102 of drawing 1 is Java for an input. Bean203 and Java for a display The user logic 205 corresponds to Bean204 and the logic 103 of drawing 1 . JSP201 for an input has the function which assigns a screen to data, and JSP202 for a display has the function which displays data on a screen. The UJI engine 206 is Java about the screen data of JSP201 for an input. It changes into data objects, such as Bean, ((1), (5), (6)). That is, the screen data of JSP201 for an input are Java for an input. It is changed into Bean203. The user logic 205 consists of two or more manipulation routines, and the UJI engine 206 determines a suitable manipulation routine based on the class of data object (Java Bean203 for an input), and a command and the external mapping file 207 ((5), (6), (7)). Java for an input which is a data object by the user logic 205 using the determined manipulation routine Bean203 is processed and they are (2) and Java for a display. It outputs as Bean204 ((3)). Java for a display Bean204 is Java for a display which it is changed into the screen data of JSP202 for a display by the UJI engine 206 ((4), (5), (6)), and is a processing result and a data object. The combination of the class of Bean is discovered from the external mapping file 207, and the components which should be displayed are determined. Thus, since this invention takes a configuration between which it is placed by the data object between HTML and logic which specify a screen, it makes it possible to connect a screen and logic with a non-dense.

[0016] Drawing 3 is drawing showing the system configuration of the 1st example of this invention. The request 301 from a client is received by the front component 302. And the request data contained in a request 301 are Java for an input which an automated analysis is carried out with the UJI engine 304, and is a data object. Bean303 is generated automatically. With the UJI engine 304, it opts for processing by the user logic 307 with reference to the command mapping 305 further from the class of data object and command which were generated from request data. The generated data object is read by the user logic 307, and it processes according to this decision, and is Java for a

display as a processing result. Bean309 is set up. Here, the UJI engine 304 refers to the page mapping 306, and is Java for a display. JSP310 for a display from Bean309, i.e., a screen expression, is determined. In addition, the template 308 which specifies a page layout is included in JSP310 for a display, and a display article is arranged according to this template 308.

[0017] Although drawing 3 showed the system configuration of the 1st example of this invention and described the outline of operation, concrete actuation of the system of the 1st example of this invention is explained using drawing 4 . The situation of the Web application the "destination notice plate" in which it is shown where a user is as for drawing 4 is shown. The user list display screen 401 which indicates the user by list, the user condition edit display 402 which edits a user's condition, the profile edit display 403 which edits a user's destination candidate, and the user registration screen 404 which registers a first time user are displayed by the web browser by the client, respectively. the log in carbon button 405 depresses in the user list display screen 401 -- having (clicked) -- it changes to the user condition edit display 402. In the user list display screen 401, if the user registration carbon button 406 is depressed, it will change on the user registration screen 404. Similarly, if each carbon button of each screens 402-404 is depressed, a screen will change like the arrow head shown in corresponding drawing 4 . At this time, each carbon button of the screen of log in 405, the user registration carbon button 406, the modification carbon button 407, and log out carbon button 408 grade is equivalent to a command for a user to demand processing from a system.

[0018] Moreover, when a user "Matsuzuka" wants to add the condition of calling it the "conference room" other than a tri-state called annual [which can be displayed as a user condition / the "seat", "a business trip", and annual / "annual"], for example, the profile edit carbon button 409 is depressed in the user condition edit display 402. Then, a screen display changes to the profile edit display 403. Here, a user "Matsuzuka" inputs a "conference room" into an "additional" column, and if the modification carbon button 410 is depressed, the profile edit display to which a new condition was added will be displayed.

[0019] Hereafter, drawing 5 and drawing 6 explain the system configuration and its outline of operation of application of drawing 4 .

[0020] Drawing 5 shows an outline of operation after the request from a client is received until processing is completed. The user list display screen 501 displayed on a web browser, the user condition edit display 502, the profile edit display 503, and the user registration screen 504 are equivalent to the screen of drawing 4 , respectively. from each screen, commands, such as a log in, user registration, profile edit, a log out, modification, a user addition, and cancellation, transmit to UJI506 -- having (continuous-line arrow head) -- Java each screen data of whose is a data object It is automatically changed into Bean505 (dotted-line arrow head). Moreover, UJI506 chooses the processing which should branch based on the combination of the transmitted command and the class of data object from processings 508-512 with reference to the command mapping 507 (after-mentioned). And a data object is passed to the selected processing. If a data object is passed, the data object will be processed in accordance with the contents of the selected processing.

[0021] Processing of a data object finishes and drawing 6 shows an outline of operation until a response is returned to a client. Processing finishes and it is Java as a processing result. Bean601 is created. And Java which the page mapping 602 (after-mentioned) was referred to in UJI506, and was created with processing results (a success, failure, completion, etc.) The page which should be displayed from combination with Bean601 is determined out of pages 501-504. Created JavaBean601 is passed to the page which should be displayed and a new page is displayed based on it.

[0022] A part of command mapping is shown in drawing 7 (**) and (b), and a part of page mapping is shown in drawing 7 (c). Command mapping consists of a class of input data object, a command, and a table that consists of processing. Moreover, page mapping consists of a class of output data object, a processing result, and a table that consists of a screen.

[0023] As shown in drawing 7 (**) and (b), the processing which should branch from the combination of the class of input data object and a command is determined by command mapping. The class of drawing 7 (**) of input data object is the same, and command mapping in case the classes of command differ is shown. Drawing 7 (**) shows that it is determined that the class of

input data object branches processing to profile modification processing when a command is "profile edit" "in the state of a user." Moreover, the classes of input data object differ and drawing 7 (b) shows command mapping when a command is the same. As shown in drawing 4 , "modification" command is used on various screens. However, even if a command is the same, the processings which should branch when a command occurs differ, respectively. When "modification" command occurs in the user condition edit display 402 of drawing 4 , processing must be branched to user status-change processing, and when "modification" command occurs in the profile edit display 403, processing must be branched to profile modification processing. By command mapping shown in drawing 7 (b), the class of input data object will branch processing to user status-change processing, if "modification" command occurs "in the state of a user", and the class of input data object can specify that it branches profile edit processing, when "modification" command occurs in "profile edit." Thus, even if command mapping of this invention has the same command in order to opt for the processing which should branch from the class of input data object, and the combination of a command, it can choose appropriately the processing which should branch from the class of input data object.

[0024] Moreover, the screen which should be displayed is determined from the combination of the class of output data object, and a processing result by page mapping shown in drawing 7 (c). Since a screen is determined in the combination of the class of output data object, and a processing result like command mapping also in page mapping, even if a processing result is the same, it is possible to choose appropriately the screen which should be displayed from the class of output data object.

[0025] In addition, command mapping and page mapping are external declaration files, and a developer can set them up freely. Thereby, the flexibility in development can be raised.

[0026] In the above, although actuation of the whole system of the 1st example of this invention was explained next, it explains more concretely about between each actuation.

[0027] Drawing 8 is Java for an input about the request data from a client. It is drawing showing the programme description for making it change into Bean (between 301 of drawing 3 , and 303). Java for a data object input which is a class name (** of drawing 8) corresponding to the specific item (LoginBean) of the HTML page 801 for data inputs, and has the identifier (name) of each input column as a property (** of drawing 8 , **') by this description when data are inputted into the HTML page 801 for data inputs Bean802 is generated.

[0028] Drawing 9 is drawing showing the programme description of the part (between 305 of drawing 3 , and 307) which opts for processing of the logic which should be performed using command mapping. The class name (LoginBean) corresponding to the specific item of the HTML page 901 for data inputs and the combination of the identifier (name) of the depressed carbon button (submit) are mapped by the command mapping 902 at the method of the user logic 903 (handler). By this, when a log in carbon button is depressed on the HTML page 901 for data inputs, the login method of the user logic 903 will be performed (** of drawing 9). Similarly, when a password change carbon button is depressed on the HTML page 901 for data inputs, the changePassword method of the user logic 903 will be performed (** of drawing 9).

[0029] Drawing 10 is drawing showing the programme description of the part (between 307 of drawing 3 , and 309) to which user logic sets the data object for a display. The user logic (handler) 1001 is a data object for a display about the value [process using a database 1002 etc. and] as a result of processing to be displayed, and a processing result. (Java Bean) It is set as 1003.

[0030] Drawing 11 is drawing showing the programme description of the part (between 309 of drawing 3 , and 310) which determines the page which should be displayed through page mapping. It is mapped in the page mapping 1102 by the view (JSP in this case) which the combination of a class name (UserBean) and the processing result set up by user logic (handler) should display. Thereby, it is Java for a display. When the processing result set as Bean1101 is succeeded, login-succeeded.jsp 1103 is chosen (** of drawing 11), this and the template 308 of drawing 3 are put together, and the display screen is determined. Similarly, it is Java for a display. When the processing result set as Bean1101 is failed, login-failed.jsp1104 is chosen (** of drawing 11), this and the template 308 of drawing 3 are put together, and the display screen is determined.

[0031] Drawing 12 is drawing showing the programme description of the part (310 of drawing 3) which outputs the display screen from JSP. JSP1201 for a display uses the tag for data acquisition,

and is Java for a display. Data are acquired from Bean1202 and the acquired data are outputted as an output HTML 1203.
[0032] Drawing 13 is drawing explaining the template in the case of arranging two or more display articles to the display screen (between 308 of drawing 3 , and 310). The user logic (handler) 1301 is Java for a display of plurality [coincidence]. Bean can be set up. For example, it is Java for banners like drawing 13 . Bean1302, Java for menus Bean1303, Java for contents It is each Java when Bean1304 is set up by the user logic 1301. JSP corresponding to Bean is each Java. Data are acquired from Bean and each display screen is created. Since the display position is specified to the template 1305 for every JSP, each created display screen is arranged based on a template 1305, and an output HTML 1306 is outputted.
[0033] As mentioned above, by the system of the 1st example of this invention, the HTTP (Hyper Text Transfer Protocol) request was changed into the data object, and was processed, and the structure which changes the data object of a processing result into a response, and is returned to a client was stated to the detail. Next, the system of the 2nd example of this invention which processes an XML (eXtensible Markup Language) file is described using the same structure. The system configuration is shown in drawing 14 and 15.
[0034] drawing 14 is drawing shown until the XML file is processed by logic (handler), when an application server receives an XML file, and drawing 15 is drawing shown until processing of the XML file finishes it as logic (handler) and a response is returned to a client.
[0035] First, when a server receives the XML instance 1401 in drawing 14 , it is XML about the XML. Data It processes with the Binding engine 1404 and a data object (Bean instance 1405) is created. Under the present circumstances, since the class of object can be specified from the received XML instance 1401, XML from which a class differs is receivable with the same engine. Here, the class of object is specified using an XML mapping file (un-illustrating, after-mentioned). Moreover, the UJI engine 1403 opts for the processing (method) which should be performed by logic (handler) from the tag of XML by referring to the command mapping 1402.
[0036] In drawing 15 , the data object for transmission (Bean instance 1502) is created completely like the time of a HTTP request by logic (handler) 1501. This is XML again. Data It is changed into XML with the Binding engine 1404, and XML1503 for transmission is outputted.
[0037] In addition, XML which generates a data object Data The Binding engine 1404 is called from a factory class. Therefore, it can respond to two or more Binding engines flexibly by preparing a factory class for every class of Binding engine.
[0038] Drawing 16 and drawing 17 show the outline of the system shown in drawing 14 and drawing 15 of operation. it is drawing shown after drawing 16 corresponds to drawing 14 and an application server receives an XML file until it was processed by logic (handler), and drawing 17 returns a response to a client, after corresponding to drawing 15 and completing processing by logic -- it is drawing having shown until.
[0039] in drawing 16 , the XML file of the order cut-form 1601 and shipping ticket 1602 grade transmits to an application server -- having (continuous-line arrow head) -- Java each screen data of whose is a data object It is automatically changed into Bean1605 (dotted-line arrow head). Moreover, in UJI1603, the class of object of an XML instance is specified using the XML mapfile 1606, and it is XMLData. The data object corresponding to XML which received with the Binding engine 1604 is created. Moreover, by UJI1603, the processing which should branch from the tag of XML and the command mapping 1607 is chosen from processings 1608 and 1609 and 1610 grades. And a data object is passed to the selected processing. If a data object is passed, the data object will be processed in accordance with the contents of the selected processing.
[0040] Java which it is as a result of processing in drawing 17 after processing finishes Bean1701 is passed to UJI1603. This Java Bean1701 is XMLData. It is changed into the XML instance 1702 for transmission with the Binding engine 1604.
[0041] Thus, it is XML about XML which received in the system of the 2nd example. Since logic (handler) is passed after changing into a data object with a DataBinding engine, when realizing the same processing as the time of a HTTP request, the method of the same logic (handler) as the logic (handler) of the system of the 1st example can be used as it is.
[0042] In the above, although the system configuration of the 1st and 2nd examples of this invention

and system behavior were explained to the detail, the structure of actuation of the system common to these examples is explained below.

[0043] The object which constitutes the system of this invention can be classified into a system, application, a session, and four steps of time scopes of a request based on the relative physical relationship on a time-axis in case each is processed. As shown in drawing 18 (**), a system 1801 exists by the one whole system, and uses from starting of an application server to termination of a server as the scope. Moreover, one application 1802 exists for every application in a system, and uses from initiation of application to termination as the scope. Every one session 1803 exists for every client, and uses from connection of a client to cutting (or time-out etc.) as the scope. And a request 1804 uses the period until even per [from a certain client] request exists, it processes the request from a client and it returns a response as a scope. The correlation between each object is shown in drawing 18 (b). Two or more applications 1802 exist in a system 1801, and the session 1803 from two or more clients linked to each application 1802 exists. And the request from a client is processed by the handler 1805 prepared beforehand. In addition, drawing 18 The request 1804 shown in (a) corresponds to one thread at the time of each object of drawing 18 (b) being performed.

[0044] Thus, when the object which constitutes a system has a gradual time scope and it processes the request from a client, it can branch from an object with a bigger scope to a small object (dispatch), and processing can be advanced. That is, if a request is transmitted to the front component 1901 from a client as shown in drawing 19 , processing will be branched to the application 1903 with which the dispatcher 1902 which exists in one system corresponds. Furthermore, application 1903 branches processing at the session 1904 corresponding to the client which has transmitted the request, and a session 1904 branches processing on the handler 1905 corresponding to a request.

[0045] in order to branch and process the request from a client on the small scope from the bigger scope in the system of this invention -- each object of application, a session, and a handler -- a user -- the following effectiveness will be acquired if custom is made extensible or possible. (In addition in the system of this example, application 1903 and a session 1904 were mounted as custom being possible in user extensible, the front component 1901, the page 1906 for a display, a handler 1905, the command map 1909, a page map 1910, and ResponseBean1908 and RequestBean1911.)

a user -- if the interface of SingleThreadModel 1912 is made to mount to the object in which custom is extensible or possible (drawing 19 (b)), actuation of the object will be restricted to coincidence up to a maximum of 1 thread. If this function is used, actuation of each scope serves as a thread safe, namely, it can avoid making two or more data processing, such as the same data editing, perform to coincidence. It is called "single thread actuation limit" to do in this way. In the above-mentioned "destination notice plate" application (drawing 4), since the same data are targetted for user modification and profile edit, it is necessary to make it thread safe. In this case, what is necessary is just to mount, as shown in drawing 20 (**). Namely, the method:user modification 2007, a method: Make a SingleThreadModel interface mount in the handler 2004 including the profile edit 2008. Since it is not necessary to make it the method:log in processing 2005, the method:log out processing 2006, method:user registration, **, especially thread safe, a SingleThreadModel interface is not made to mount in handler 2004' and 2004" here. Similarly, it is SingleThreadModel when you want to make it thread safe also in a session and application. What is necessary is just to make an interface mount (2002', 2003'). The method of description in the case of mounting a SingleThreadModel interface is SingleThreadModel to session 2003' in this drawing that what is necessary is just to carry out like drawing 20 (b). The case where an interface is mounted is shown.

[0046] moreover, a user -- pretreatment (preprocessor) and an after-treatment (postprocessor) interface can be made to mount in the application and the session which are the object in which custom is extensible or possible, and a handler object Thereby, processing can be added before and after activation of acting-before-the-audience processing of each object. That is, as shown in the sequence diagram of drawing 21 , pretreatment (** of drawing 21) of acting-before-the-audience processing (handler method) (** of drawing 21) of the request of a client and after treatment (** of drawing 21) can be added, and pretreatment (** of drawing 21) to a session (** of drawing 21), after treatment (** of drawing 21), pretreatment (** of drawing 21) to application (** of drawing 21), and after treatment (** of drawing 21) can be added similarly.

[0047] This function is used, for example, it is acting-before-the-audience processing of the handler method of logic by pretreatment (** of drawing 21). (** of drawing 21) It is possible to prepare the check routine which makes a judgment by status management beforehand in front, and skips processing of a handler method to it. Moreover, in after treatment (** of drawing 21), when an error comes out by pretreatment (** of drawing 21), or acting-before-the-audience processing (** of drawing 21), the error handling for recovering the error can be prepared. Even when an error does not come out, these can also be used in order to describe common after treatment. Similarly, common pretreatment to one client and after treatment can be described about each session, and application can describe common pretreatment to two or more clients, and after treatment. Since each pretreatment and after treatment can be separately mounted in the object of each scope, they can perform fine control.
[0048] As mentioned above, if the structure of actuation of the system of this invention is used, the effectiveness that a thread safe can be realized or pre-decision of processing and error processing can be flexibly performed now by adding pretreatment and after treatment to the processing in each scope can be acquired.
[0049] Next, the structure for requesting to a browser from the handler of the logic in the application server of this invention is explained.
[0050] or [continuing processing A by asking a user (client) whether continue processing A in Web application, based on the reply, for example, since a certain thing happened although the request of Processing A has been sent from the client] -- or it may be said that it is made to perform different processing Thus, he is a user in the middle of processing. (client) When an inquiry is performed and it processes based on the result of this inquiry, general Web application system is performing by the sequence as shown in drawing 22 (**). If the request (** of drawing 22 (**)) of Processing A is transmitted to a server, in a server, it will be received and a handler 1 will begin processing. Then, although Processing A all is not ended, it asks a user whether processing A is continued (** of drawing 22 (**)). A screen like 2701 is displayed on a client and a user returns reply [which / of "yes" and "no"] to it (** of drawing 22 (**)). Based on it, a handler 2 begins processing.
[0051] Thus, processing of a handler may have to be interrupted for general Web application even if Processing A is not completed altogether. Moreover, it needs to be mapping defined for the intermediate processing intermediate treatment as which the reply of a user specified as opposed to [what question] cases, such as "yes" and no ["no"], are "yes" and no ["no"].
[0052] Then, the structure which requests to a browser from the handler of the logic in an application server which is described below was introduced into the system of this invention, and general Web application system has been improved.
[0053] As shown in drawing 23 (**), in the system of this invention, in a server, a thread buffer object (synchronizer) is prepared before a handler call, and actuation of a handler is supervised. Thereby, if the request (** of drawing 23 (**)) of Processing A is transmitted to a server, in a server, a synchronizer will receive, a handler 1 will be passed (** of drawing 23 (**)), a handler 1 will start, and processing will start. Then, although Processing A all is not completed, when questioning [whether processing A is continued and] a user, a handler 1 calls the method which calls back a client (** of drawing 23 (**)). In a synchronizer, the method of a handler is returned as a return value to a browser (client) (** of drawing 23 (**)). A screen like 2201 is displayed on a client and a user returns reply [which / of "yes" and "no"] to it (** of drawing 23 (**)). It is received by the synchronizer and it is passed to a handler 1 (** of drawing 23 (**)). Then, a handler 1 continues processing. It becomes unnecessary thus, to interrupt processing of a handler for a thread buffer object being prepared. Moreover, also in description of the program for realizing a server, since a conditional-branching sentence can describe in one handler to having to indicate a handler 1, the handler 2 which receives the event to the return value of a handler 1 as this invention shows to drawing 23 (b) and (c) as general Web application shows to drawing 22 (b), it is very easy in programme description.
[0054] In addition, although HTTP, XML, etc. were targetted as an input from a client in the example of this invention explained by the above, as long as it is a request from a client, what kind of thing may be used, and at the example of these this inventions, it is Java about the request of a client. Although changed into a data object called Bean, other data objects are sufficient.

[0055] Invention concerning the example explained by the above changes the input from a client into a data object, processes it, is characterized by to change into the response to a client the data object obtained as a processing result, and to transmit, and is characterized by to classify processing [in / a server] into a time scope, to branch to a small thing from what has a big scope, and to advance processing.
[0056] Next, the further example of this invention is explained.
[0057] In case the example explained from this develops the application server which performs Web application, as it separates the definition (view) of the appearance of a screen, and the data embedded all over the screen and mounts them, it raises both the development effectiveness and maintainability of Web application.
[0058] The display in Web application is divided and prepared for a display data object and the definition of the appearance (view) of a screen in this example. At this time, a display data object has the indicative data itself, and creates it as a structure class which is an interface showing DS (suppose that this DS is called a "model" in this example), such as table (table) structure and the tree structure, and mounted the thing about that indicative data. Moreover, the definition of a view is described using JSP (Java Server Pages) in this example (you may make it HTML only describe the definition of a view).
[0059] If the definition of this display data object and a view is performed in a server, the HTML sentence which assigns an indicative data to the predetermined location of a screen with the activation engine corresponding to the interface mentioned above (it assigns) will be generated. Then, if this HTML sentence is sent to a client, the screen display which corresponds by the web browser with which a client is equipped will be performed.
[0060] In addition, in order to distinguish the example which is explained from this and which was mentioned above from other already explained examples, suppose that "the 3rd example" is called.
[0061] Drawing 24 is explained first. This drawing shows the system configuration of the 3rd example of this invention.
[0062] The model object 3001 is an object held in order to pass the data which held the data which should be displayed on the client acquired from back-ends, such as a database formed in the server side, and were sent from the client to a back-end, and mounts the model interface 3002 mentioned later. In addition, the model object 3001 is possible also for preparing separately as model object 3001a for a display, and model object 3001for input b, and can also be prepared as a single model object 3001 which shares the object for a display, and the object for an input.
[0063] The model interface 3002 is an interface showing the structure of a specific model, for example, the interface according to the target DS is prepared [tree structure / the object for table models, and] like the object for tree models etc. about a table structure.
[0064] The framework engine 3003 takes out the data currently held at model object 3001a for a display based on the model interface 3002 mounted in model object 3001a for a display, and generates a HTML sentence according to a definition of JSP3004 for a display. Moreover, the data inputted to the browser 3005 are analyzed and model object 3001b for an input is passed.
[0065] The appearance (view) of the screen when expressing an indicative data in HTML is defined, and JSP3004 for a display is described by JSP which introduced the original tag.
[0066] a client is equipped with a browser 3005, and it comes out and performs a screen display according to HTML generated with the framework engine 3003 based on the definition of JSP3004 for a display.
[0067] Although the front component 3006 receives the request emitted from a browser 3005, passes it to the framework engine 3003 and uses the JSP component or a servlet, as long as it can receive a HTTP request and can call the framework engine 3003 according to the receiving contents, what kind of thing is sufficient as it.
[0068] Next, about the model interface 3002 of drawing 24 , the table model interface showing the model of a table structure is mentioned as an example, and is further explained to a detail.
[0069] Drawing 25 is drawing showing the example of declaration of a table model interface. It roughly divides and the table model interface is defined in this example by the method for substitution (to back-end) of the method for acquisition of the method for data acquisition, and a data class, and data.

[0070] If these methods are explained further, the method (getColumnCount) which acquires the number of trains of a table as a method for data acquisition, the method (getRowCount) which acquires the line count of a table, and the method (getValueAt) which acquires the data of each cel of a table are used for the definition of drawing 25 .
[0071] The method (getColumnClass and getRowClass) which acquires respectively the character string which specifies the "class" by which each of a train and a line is defined as a method for acquisition of a data class is used for the definition of drawing 25 . This "class" is "cls" of the renderer tag which it is used in order to define the formats (for example, a display position, a format called the magnitude of a font) of the method of presentation about the data belonging to each train or each line, respectively, and this mentions later. An attribute is shown.
[0072] The contents (location on the value of a cel and the table of that cel) of the request from a client are passed to a back-end in the data format depending on the DS shown with this table model by calling the method (setValueAt) for substitution of data by processing (after-mentioned) of the model update corresponding to this table model.
[0073] Next, signs that a HTML sentence is generated in the 3rd example of this invention are explained using drawing 26 . This drawing shows that the framework engine 3003 generates a generation HTML sentence (C) based on the JSP source (B) equivalent to JSP3004 for a display, if model object 3001for display a (A) which realizes the table model interface with which the declaration shown in the framework engine 3003 for table models shown in drawing 24 at drawing 25 is made is given. In addition, the line number is given to each head of the sentence of drawing 26 (B) and drawing 26 (C) for the facilities of explanation.
[0074] (B) JSP source of drawing 26 is explained.
[0075] In drawing 26 (B), the 1st line specifies the correspondence relation between this JSP source and a model object (A).
[0076] It is "id= when the 1st line is explained further... A name in case a corresponding model object is referred to in this JSP source is specified as ", and it is "cls=... The class name of the model object to which "corresponds with this JSP source is specified.
[0077] Moreover, it sets to the 1st line and is "request=... The life time within the framework engine 3003 of a model object (A) is specified as ". After this sends a generation HTML sentence (C) to a client and makes a screen display perform to a browser 3005, it takes into consideration the case where the request according to this model object is answered from a client. That is, when true is specified, in order to enable it to substitute the contents of such a request for a model object immediately, a model object (A) is held also even for after generation of a HTML sentence in the storage region in the framework engine 3003. On the other hand, when false is specified, after generation of a HTML sentence is not held in the storage region in the framework engine 3003 of a model object (A).
[0078] After the 2nd line of drawing 26 (B), it is the part in which the definition of the appearance of the screen for displaying a table model is made using the tag, and this part is named the "view" generically.
[0079] Description of a view is performed using a view tag. The initiation tag of the view tag for a table model is shown in the 2nd line (<uomf:table>), and the renderer for some tables is described between the termination tags (</uomf:table>) of this view tag for table models of the 2nd line and the 18th line.
[0080] A renderer defines what kind of the method of presentation is used for the display of the data in the appointed display location. In the description applied to the 17th line from the 3rd line of drawing 26 (B), a total of five (e) is made for the definition of the renderer for a table model from (a). A thing for (a) to make a table tag (<table> and </table>) generate, when the definition of these table renderers is explained briefly, What (b) defines the means of displaying of the line of a table by, the thing by which (c) defines the means of displaying of the usual train in a table, What (d) defines by the means of displaying of a cel (header cel) in which the header of each train in a table is shown, and (e) define the means of displaying about the possible cel of the data input (renewal of data) in a table.
[0081] The initiation tag (<uomf:tableRenderer>) of the renderer tag for table models is described by each renderer at the first line, and the termination tag (</uomf:tableRenderer>) corresponding to this

is described by the line of the last. And the method of presentation of data is defined by description of the line inserted into the initiation tag of this renderer tag, and the termination tag. Each element which constitutes the definition about the method of presentation of this data is called the renderer element.
[0082] The assignment about "type" and "cls" is made with a renderer tag. These are for pinpointing the location on the screen which displays with the method of presentation defined as the renderer. These contents of assignment are beforehand specified according to the class of model to treat.
[0083] With the table model in the gestalt of this operation, table (front whole), row (line), or column (train) is specified, and the location of the table which displays with the method of presentation defined as the renderer by these is pinpointed in "type."
[0084] For "cls", "type" mentioned above in the table model in the gestalt of this operation is row. Or when specified as column, it can specify (there may be no assignment). This is being interlocked with both the methods of getRowClass in the example of declaration of the table model interface mentioned above (drawing 25), and getColumnClass, and in case a train is displayed [for example,], the getColumnClass method of a table model interface is called. And the definition of the renderer of a view is referred to, "type" is column and the definition of the method of presentation in the renderer corresponding to the return value about the method to which "cls" was called is used for generation of a HTML sentence as the method of presentation for the display of the train. In addition, when there is no return value about the called method (it is null), the definition of the method of presentation about a renderer without assignment of "cls" is used.
[0085] In a table model [in / in "cls" / the gestalt of this operation], header or editable is specifically specified.
[0086] Although the definition of the method of presentation in a renderer is described using the tag of well-known HTML, a special tag called a renderer element tag is introduced here. It sets to drawing 26 (B) and is <uomf:children>. And <uomf:value> is a renderer element tag. <uomf:children> makes the actuation which develops the definition of the method of presentation in other renderers in the location of this tag perform in a framework engine, and <uomf:value> makes the actuation on which it is made to display the value (indicative data) acquired from the model interface in the location of this tag perform in a framework engine.
[0087] More detailed actuation of these tags is beforehand prescribed by the class of model to treat. the table model in the gestalt of this operation -- <uomf:children> The means of displaying defined by the renderer (renderer whose assignment of "type" is row) about a line when assignment of "type" in the renderer tag mentioned above is table (front whole) and this tag appears is developed and inserted in this location. When this tag appears in the renderer whose assignment of "type" is row (line), developing and inserting in this location the means of displaying defined by the renderer (renderer whose assignment of "type" is column) about a train is specified. Moreover, the assignment of "type" of <uomf:value> is usable only in the renderer which is column, and inserting in this location the value acquired by the getValueAt method currently used in the example of declaration of a table model interface (drawing 25) is specified.
[0088] Hereafter, signs that the generation HTML sentence of (C) is generated by the framework engine are explained, referring to drawing 26 .
[0089] First, a framework engine recognizes that the model object of (A) mounts the table model interface, and starts the framework engine for table models. And getColumnCount and getRowCount of this table model interface Both methods are called and the number of trains and line count of a table to display are recognized. Consequently, it is recognized that the table of three-line three trains is displayed here.
[0090] Next, it sets in the JSP source of (B) and "type" is table. The definition of the table renderer of (a) specified is referred to first, and the 1st line of the generation HTML sentence of (C) is generated.
[0091] The renderer element tag <uomf:children> is described by the definition (the 4th line of (B)) of the table renderer of (a) here. "type" in the renderer tag of (a) is table. Since it is specified, expansion of a definition of the table renderer about a line, i.e., the table renderer of (b) in the JSP source of (B), is performed, and the 2nd line of the generation HTML sentence of (C) is generated.
[0092] Here, "type" is row. Since it is specified, expansion of a definition of the table renderer about

a train is performed. [in / the renderer element tag <uomf:children> is described by the definition (the 7th line of (B)) of the table renderer of (b), and / the renderer tag of (b)] However, since three table renderers (renderer whose assignment of "type" is column) about the train (c), (d), and (e) are described by the JSP source of (B), whether the definition of the table renderer of a throat is developed among these poses a problem.

[0093] At this time, acquisition of the value of the cel of each train of the table to display of the 1st line and the call of the getColumnClass method of a table model interface are also performed from the model object of (A) with the framework engine. The return value about the method which the values of the cel acquired here are the three character-string data a "name of article", a "unit price", and the "number", and was called by these processings was header also about which [these / three] value.

[0094] Then, based on the return value about this method, the definition of the table renderer of (d) in the JSP source of (B) is chosen, and a framework engine performs expansion of that definition. Consequently, the HTML sentence (the 3rd of the generation HTML sentence of (C), the 4th, the 5th line) which shows the means of displaying of the 1st line of the table to display is generated based on the contents of a definition of the table renderer of (d) (the 13th line of (B)). Here, since the renderer element tag <uomf:value> is contained in the definition of the table renderer of (d), the value of each cel acquired from the model object is inserted in this part.

[0095] renderer element tag <uomf:children> in the definition (the 7th line of (B)) of the table renderer of (b) which came out of so far and which was performed previously about -- expansion is completed, based on description of a definition of the degree following the tag, the 6th line of the generation HTML sentence of (C) is generated, and generation of the HTML sentence for the display of the 1st line of a table is completed.

[0096] Next, generation of the HTML sentence for the display of the 2nd line of a table is started, expansion of a definition of the table renderer about a line, i.e., the table renderer of (b) in the JSP source of (B), for the second time is performed, and the 7th line of the generation HTML sentence of (C) is generated.

[0097] Here, expansion to the definition of the table renderer about the train of a renderer element tag <uomf:children> is performed to the definition of the table renderer of (b) like last time.

[0098] At this time, with a framework engine, from the model object of (A) Acquisition of the value of the cel of each train of the table to display of the 2nd line and the call of the getColumnClass method of a table model interface are also performed. By these processings The return value about the method which the values of the cel acquired here are "XV-5080", "198,000", and the three data "" (null), and was called the value about the first two cels -- null -- it was data and the value about the last cel was editable.

[0099] Then, a framework engine performs expansion of a definition of the table renderer of (c) in the JSP source of (B) based on the return value of this method about the first two cels. Consequently, the HTML sentence (the 8th and the 9th line of the generation HTML sentence of (C)) which shows the means of displaying of eye the two-line train [one train] of a table and eye two-line trains [two trains] to display is generated based on the contents of a definition of the table renderer of (c) (the 10th line of (B)). Here, since the renderer element tag <uomf:value> is contained also in the definition of the table renderer of (c), the value of each cel acquired from the model object is inserted in this part.

[0100] Furthermore, a framework engine performs expansion of a definition of the table renderer of (e) in the JSP source of (B) based on the return value of this method about the last cel mentioned above. Consequently, the HTML sentence (the 10th line of the generation HTML sentence of (C)) which shows the means of displaying of eye two-line trains [three trains] of the table to display is generated based on the contents of a definition of the table renderer of (e) (the 16th line of (B)).

[0101] renderer element tag <uomf:children> in the definition (the 7th line of (B)) of the table renderer of (b) which came out of so far and which was performed previously about -- expansion is completed, based on description of a definition of the degree following the tag, the 11th line of the generation HTML sentence of (C) is generated, and generation of the HTML sentence for the display of the 2nd line of a table is completed.

[0102] The flow by which (the HTML sentence for the display of the 3rd line of a table) is generated

from the 12th line of the generation HTML sentence of (C) to the 16th line is the same as that of the HTML sentence for the display of the 2nd line of the table mentioned above (from the 7th line of the generation HTML sentence of (C) to the 11th line).
[0103] renderer element tag <uomf:children> in the definition (the 4th line of (B)) of the table renderer of (a) which came out of so far and which was performed previously about -- expansion is completed and the 17th line of the generation HTML sentence of (C) is generated based on description of a definition of the degree following the tag. In this way, generation of the HTML sentence for the display of the table shown in (C) is completed.
[0104] In addition, as mentioned above, the generated HTML sentence is addressed and sent out to a client. Then, since it enables it to specify easily the thing about which model object that request is when the request corresponding to this HTML sentence has been sent from the client, the input tag (<input type=hidden>) of the so-called hidden attribute with which input field are not displayed by the client is added and sent out to this generated HTML sentence by the browser. As meaning ID (identifier) is described for every view used for generation of the character string which shows the class of model to this input tag, and a HTML sentence, these data are made to be contained in that request automatically.
[0105] Along with the flow chart which shows the processing explained [by] since the framework engine 3003 for table models shown in drawing 24 generated the HTML sentence for displaying a table on a client by the browser 3005 to drawing 27 , drawing 28 , and drawing 29 , it explains further. In addition, suppose that this processing is called "processing at the time of a display."
[0106] First, the framework engine 3003 reads JSP3004 for a display, and detects the view tag contained in JSP3004 for a display. Then, each renderer contained in the view shown with the view tag is classified based on "type" specified with the renderer tag, and "cls", and it registers with the predetermined location of a buffer. Furthermore, after finishing this registration, the framework engine 3003 starts the processing which met drawing 27 , drawing 28 , and the flow chart shown in drawing 29 , acquiring data from the model object 3001 specified as the view tag. In addition, suppose that the processing so far is named "analysis processing of a view tag" generically.
[0107] The whole control processing flow at the time of the display performed with the framework engine 3003 for table models is shown in drawing 27 .
[0108] Initiation of the processing shown in drawing 27 generates the meaning ID of a view first (S3101). The meaning ID of a view is meaning ID, for example, as mentioned above, as shown in "uji.model.00001", for every view of JSP3004 for a display used for generation of a HTML sentence, it should just generate what combined the figure ("00001" parts) of consecutive numbers which are different at a common prefix (prefix) (part of "uji.model."), and every HTML sentence generation. It is used in order that different meaning ID for every view of this may enable it to identify the thing about which model object the received request is, respectively when the display about two or more model objects is performed by the client and a server receives the request about those each, and in a client, the request containing ID corresponding to a model object is generated.
[0109] Then, each renderer registered into the buffer which the framework engine 3003 has by processing mentioned above to "type" is table. What is specified is acquired (S3102).
[0110] Then, it is table about level. It carries out and is row about child level. Renderer display processing when carrying out is performed (S3103). A renderer display process is the processing shown in drawing 28 with the flow chart, and the detail is mentioned later.
[0111] The HTML sentence for displaying a table by processing of S3103 mentioned above is generated. Then, the framework engine 3003 generates the input tag of the hidden attribute mentioned above, and adds it to this HTML sentence (S3104). Let the input tag generated here be the following, for example.
<input type=hidden name="uji.model" value=" uji.model.00001" <input type=hidden name="uji.model.00001" value="table">> -- here, "uji.model.00001" is the example of the meaning ID of the view generated previously. By adding these input tags to a HTML sentence, such information comes to be included in the request corresponding to this HTML sentence, and the correspondence relation between a request and the model object 3001 becomes clear.
[0112] Next, renderer display processing which is performed in S3103 of drawing 27 and which is shown in drawing 28 with a flow chart is explained.

[0113] In addition, in subsequent explanation and a drawing, it adds to the notation (for example, <umof:children>) of the renderer element tag used until now, and is uomf. : Suppose that the notation (for example, <children>) which omitted the prefix is also used together.
[0114] First, if it is investigated what level when this renderer display process is called was, it is judged whether the element which defines the means of displaying in the renderer by which this level is specified as "type" is yet left behind (S3201) and a judgment result becomes Yes, the following one element (renderer element) will be acquired (S3202), and processing will progress to S3203. On the other hand, if the judgment result of S3201 becomes No, this renderer display processing will be completed and it will return to the original processing.
[0115] Then, it is judged what kind of thing the element acquired by processing of S3202 is (S3203, S3206, S3208, S3210).
[0116] Consequently, if the acquired element is a view tag (the judgment result of S3203 is Yes), analysis processing of the view tag mentioned above will be performed to this view tag (S3204), processing of drawing 27 mentioned above will be performed about this view tag after that (S3205), and, as for after termination of this processing, processing will return to S3201.
[0117] This processing of S3204 and S3205 is for making it correspond, when still more nearly another model exists in the cel which is performed when description of another view during description of a certain view is made (the view nests), and has a table model.
[0118] If the element acquired by processing of S3202 is the renderer element tag of <children> (the judgment result of S3206 is Yes), it will be investigated what child level when this renderer display process is called was, the display process about this child level will be performed (S3207), and processing will return to S3201 after termination of this processing. The display process in case child level is row is shown to drawing 29 (A) by the flow chart, and the display process in case child level is column is shown to drawing 29 (B) by the flow chart. These display processing is explained later.
[0119] Moreover, if the element acquired by processing of S3202 is the renderer element tag of <name> (the judgment result of S3208 is Yes), processing which performs naming to the element behind used in a request will be performed (S3209), and processing will return to S3201 after termination of this processing.
[0120] Here, the renderer element tag of <name> is explained. This tag makes a framework engine generate the identifier according to a predetermined regulation.
[0121] In this example, this identifier is generated according to the regulation shown in (A) of drawing 30 . What is necessary is here, for a model proper location to be a character string for expressing the self location in a certain model to a meaning, for example, just to generate a character string like "2_3" etc., if eye two-line trains [three trains] is expressed in a table model. If the example of generation of the meaning ID of the view mentioned above ("uji.model.00001") is diverted as it is at this time, the identifier generated according to the renderer element tag of <name> at this time will become as shown in (B) of drawing 30 .
[0122] By performing naming under such a regulation, the location in a model object and its model can be pinpointed from this identifier.
[0123] The example of a definition of the renderer which used the renderer element tag of <name> is shown in drawing 31 . If the description of a definition shown in this drawing is explained, it will be inserted in the cel in which updating in a table is possible as a value before the value acquired from the model object updating by <uomf:value/>, and will be displayed by the client. Here, a request will be transmitted if the value of this cel is updated in a client. At this request, it is used as a reference name for referring to the value of the cel after the identifier generated by <uomf:name/> updating. Since the location in a model object and its model can be pinpointed from this reference name under the regulation of naming mentioned above with a table model engine by carrying out like this, two or more updating data can be included during one request generated by the client, and two or more updating data about a different model object can also be further included in one request.
[0124] Moreover, it is <uomf:name> in a nested view which was mentioned above. When the renderer element tag is described The above-mentioned view meaning ID is generated combining a consecutive figure etc. to what made the prefix the identifier (for example, identifier shown in (B) of drawing 30) generated on the outside of a nest like the example shown in (C) of drawing 30 . What combined the model proper location in the nested view with the view meaning ID is generated as an

identifier about a renderer element tag. Under the regulation of this naming, recognizing existence of a nest and the succession relation of a view and the model concerning that nest can be specified from the generated identifier.
[0125] It returns to explanation of drawing 28 .
[0126] The element acquired by judgment processing of S3202 is <value>. If it is a renderer element tag (the judgment result of S3210 is Yes), the value corresponding to the current location in this table model will be acquired from a model object (S3211), and processing will return to S3201 next.
[0127] If which tag which the element acquired by judgment processing of S3202 mentioned above on the other hand is a so-called thing (all of the judgment result of S3203, S3206, S3208, and S3210 are No), this acquired element will be displayed on a generation HTML sentence as it is (S3212), and processing will return to S3201 next.
[0128] Processing to the above is renderer display processing.
[0129] Next, the display process about the child level performed in S3207 of the renderer display process mentioned above is explained. it mentioned above -- as -- child level -- row it is -- the display process at the time is shown to drawing 29 (A) by the flow chart, and the display process in case child level is column is shown to drawing 29 (B) by the flow chart.
[0130] First, the flow chart of drawing 29 (A) is explained.
[0131] First, the value of Variable row is set to 0, and processings from S3302 to S3304 are repeated until the value of this variable row exceeds the return value (namely, line count of a table) of a getRowCount() method (S3301).
[0132] Then, the line class (getRowClass()) about the line shown with the value of Variable row is acquired from the model object 3001 (S3302).
[0133] Next, the acquired line class is specified as "cls" from each renderer registered into the buffer which the framework engine 3003 has, and "type" is row. The renderer specified is acquired (S3303).
[0134] Here, it is row about level. Renderer display processing when carrying out and setting child level to column is performed (S3304). The renderer processing performed here is shown in already explained drawing 28 .
[0135] Then, the value of the result of having added 1 to the value of Variable row is anew assigned to Variable row, and processing returns to S3301 (S3305). If the conditions which the value of Variable row mentioned above are reached, this processing will be completed, and it returns to the original processing.
[0136] Next, the flow chart of drawing 29 (B) is explained. The same processing as drawing 29 (A) which mentioned this processing above fundamentally is performed.
[0137] First, the value of Variable column is set to 0, and processings from S3402 to S3404 are repeated until the value of this variable column exceeds the return value (namely, the number of trains of a table) of a getcolumnCount() method (S3401).
[0138] Then, the train class (getColumnClass()) about the line shown with the value of Variable row is acquired from the model object 3001 (S3402).
[0139] Next, the renderer by which the acquired train class is specified as "cls", and "type" is specified as column is acquired from each renderer registered into the buffer which the framework engine 3003 has (S3403).
[0140] Here, renderer display processing when setting level to column and making child level nothing (null) is performed (S3404). The renderer processing performed here is also shown in already explained drawing 28 .
[0141] Then, the value of the result of having added 1 to the value of Variable column is anew assigned to Variable column, and processing returns to S3401 (S3405). If the conditions which the value of Variable column mentioned above are reached, this processing will be completed, and it returns to the original processing.
[0142] Processing to the above is display processing.
[0143] In addition, after this display process is completed, the framework engine 3003 performs storage region management processing shown in drawing 32 with reference to the description (the example of drawing 26 (B) the 1st line) the correspondence relation between the JSP source of JSP3004 for a display and the model object 3001 is indicated to be. That is, it is request by this

description. If the contents of the model object 3001 currently held at the storage section which the framework engine 3003 has if it is judged whether it is set as true (S3501) and this judgment result becomes Yes are continued also after that (S3502) and this judgment result becomes No, the storage region of the model object 3001 in the storage section which the framework engine 3003 has will be released (S3503).

[0144] Next, in the system shown in drawing 24 , the actuation which the framework engine 3003 receives the request emitted from a browser 3005, and updates the model object 3001 is explained. In addition, suppose that this actuation is called "actuation at the time of a request."

[0145] Drawing 33 is drawing explaining the outline of the actuation at the time of the request in the system shown in drawing 24 .

[0146] First, the browser 3005 by which the client is equipped with the HTML sentence containing the usual <(it is not hidden attribute) input> tag generated with the framework engine 3003 by reception and the client is displaying the table based on the HTML sentence. If the data corresponding to the this <input> tag are inputted into a client here, a browser 3005 will transmit the request by HTTP containing this data to a server side.

[0147] If this HTTP request is received, the front component 3006 of a server will give the directions for starting the framework engine 3003, and will pass a request to the framework engine 3003.

[0148] With the framework engine 3003, the received request judges first the thing about which model it is. This decision is <input> of the hidden attribute mentioned above. The information about the class of model contained in a request by operation of a tag is used. Here, if this request is judged to be a thing about a table model for example, the engine for table models will be chosen and started.

[0149] Then, similarly the framework engine 3003 is <input> of a hidden attribute. The model object 3001 which should substitute the data contained in this request from the meaning ID of a view which will be contained in a request by operation of a tag is specified.

[0150] And if the specified model object 3001 is held in the storage region in the framework engine 3003, and the data contained in the request to the thing currently held are substituted and it is not held, the model object 3001 is generated anew and the data is substituted.

[0151] In addition, in order to substitute data to the model object 3001, what is necessary is just made to update the model object 3001 by changing into the updating method according to a model the data which should be substituted, and calling the interface method (it being a setValueAt method if the table model interface 3002 shown in drawing 25 is updated) which makes a model object update.

[0152] A flow chart shows the contents of processing of the control processing at the time of the request which is performed with the framework engine 3003 and which was mentioned above to drawing 34 .

[0153] Drawing 34 shows this whole control processing flow as (A), and shows as (B) the model update processing for table models performed in the middle of this control processing.

[0154] It already explained that the reference name of the data based on the regulation of naming mentioned above was shown to the request containing the data inputted according to the display based on the HTML sentence which was generated by the browser 3005 of a client, and which was generated by the display process mentioned above. Then, the framework engine 3003 will take out first the whole of the reference name contained in the request, if a request is received. And one of the taken-out reference names is substituted for Variable key (S3601).

[0155] If it is judged whether the value of this variable key is a thing showing the model which can be treated with the framework engine 3003 here (S3602) and this judgment result becomes Yes, it will be judged whether the model object 3001 corresponding to that variable key is held continuously at the storage section which the framework engine 3003 has (S3603).

[0156] If the model object 3001 currently held at that storage section if this judgment result of S3603 becomes Yes is acquired (S3604) and this judgment result of S3603 becomes No, the model object 3001 will be created anew (S3605). And model update processing is performed to this model object 3001 (S3606).

[0157] When the judgment result of S3602 is No after completing model update processing or, another reference name taken out from the request is set as Variable key, and processing is repeated

after that until it is carried out about all the reference names with which the processings from S3602 to S3606 mentioned above were taken out (S3607).
[0158] Next, the model update processing for table models which is performed in the processing of S3606 mentioned above and which is shown in drawing 34 (B) is explained.
[0159] First, the value (namely, reference name of the updating data contained in a request) of Variable key is decomposed, and the line (row) on a table model and the location of a train (column) are acquired from the model proper location which constitutes the identifier and which was mentioned above (S3701).
[0160] And by calling the setValueAt() method of the model object 3001 and passing a location with the row and column of the acquired table model, and the value of the updating data referred to by this reference name to that method, the model object 3001 is updated (S3702) and processing returns to drawing 34 (A) after that.
[0161] When the framework engine 3003 performs processing to the above, renewal of the model object 3001 based on the request emitted from a browser 3005 is performed.
[0162] The example of the client/server system which carries out the 3rd example of this invention explained by the above is shown in drawing 35 .
[0163] A server 3010 is equipped with the model framework processing section 3011, the Web server section 3012, and a back-end 3013.
[0164] The model framework processing section 3011 performs the function equivalent to the framework engine 3003 in drawing 24 .
[0165] The Web server section 3012 performs the function to send the HTML sentence generated in the model framework processing section 3011 to a client (3020a, 3020b, 3020c, --), and the function equivalent to the front component 3006 which receives the land engine failure from a client (3020a, 3020b, 3020c, --), and is passed to the model framework processing section 3011.
[0166] A back-end 3013 operates the data stored in the database 3014, and performs transfer of the model framework processing section 3011 and data using the model object 3001.
[0167] A client (3020a, 3020b, 3020c, --) equips a browser 3005, and performs transmission addressed to the generation and the server 3010 of a request corresponding to the display of a screen based on the HTML sentence sent from a server 3010, and the input to the entry form contained in the display screen.
[0168] Next, the further example of this invention is explained.
[0169] In case the example explained from this develops Web application It is what makes available description which calls the definition of the processing script which is logic, without describing directly to the module which defines the appearance of a screen. When components-ization of the logic expressed by the definition of a processing script is attained and separation with logic and a screen definition progresses, the reusability of logic is raised and the development effectiveness of Web application is raised.
[0170] In addition, in order to distinguish the example which is explained from this and which was mentioned above from other already explained examples, suppose that "the 4th example" is called.
[0171] First, drawing 36 is explained. By performing contents transform processing later mentioned to the descriptive text which this drawing shows signs that a script is generated, according to the 4th example of this invention, and is shown in (A) shows that the HTML sentence containing the script shown in (B) is generated.
[0172] Drawing 36 (A) and the contents of description of (B) perform the processing script which performs processing on which the warning screen which notifies that click actuation was made is displayed in the display by the browser according to the click actuation made to the form components for a certain input. In addition, the line number is given to each head of the sentence of (A) of drawing 36 , and (B) for the facilities of explanation.
[0173] In order to make the processing mentioned above perform to a browser according to the development technique of the conventional Web application Description of the property which becomes a HTML sentence, i.e., the conditions which perform the call of a script part, as shown in drawing 36 (B) (drawing 36 (B) the 8th-line description), The definition (drawing 36 (B) the 1st line to 6th-line description) of actuation of the processing script itself was described directly, and in order to show these both relation further, the identifier (the example of drawing 36 (B) action123())

of a processing script needed to be described to those both sides. Therefore, since it will be necessary to check whether the correction to one of these has affected another side with reference to both after all even if correction is needed for these either for a certain reason, the problem was in maintainability. Moreover, for this reason, the definition of a processing script of operation and the part which performs the call of that processing script will be described in the same component after all, and the reusability of a definition of a processing script of operation was also low.
[0174] The descriptive text using various kinds of tags as shown in this drawing (A) is created, and it is made to, make the HTML sentence containing the script shown in (B) by performing contents transform processing to this descriptive text generate automatically in this 4th example on the other hand. A name is not specified as a processing script and the property (description which starts in "onclick=" of the 8th line of this drawing (B), and specifies a processing script name) which shows the conditions which start a processing script is not further described by this drawing (A), either.
[0175] First, a required concept is explained when explaining this drawing (A).
[0176] Drawing 37 shows the configuration of the object used in the 4th example of this invention.
[0177] The component (ScriptComponent) object 4001 is an object which becomes the generating origin of the event to the object currently shown by the browser all over the screen, and the <input> tag (ValidInputTag object 4011) in the example of drawing 36 (A) is called a component tag. The component tag has the action tag in the interior, and assignment of the event for calling a processing script which is the contents of the property described by the action tag is performed.
[0178] The container (ScriptContainer) object 4002 is an object which outputs the processing script which management of the component object 4001 and the action object 4003 have, and the <form> tag (ValidFormTag object 4012) in the example of drawing 36 (A) is called a container tag. In addition, the container object 4002 has inherited the component object 4001, and oneself may become the generating origin of an event.
[0179] The action (ScriptAction) object 4003 is an object by which the definition of the processing which is logic is made in a script, and the <action> tag in the example of drawing 36 (A) is called an action tag. In addition, although an action tag has a custom-made action tag (CustomActiontag object 4013a) and a mire cushion tag (MyActiontag object 4014b), this detail is mentioned later.
[0180] The script call section (ScriptCaller) object 4004 is an object which determines the method of calling each processing script when calling a processing script according to an individual according to generating of an event, and the interpretative method of a return value.
[0181] Next, drawing 38 is explained. This drawing is drawing explaining the outline of contents transform processing. Signs that the descriptive text shown in drawing 36 (A) is changed hereafter are explained referring to drawing 38 .
[0182] First, the contents of description of the <form> tag which is the container object 4102 described by the 1st line of (A) are memorized by the storage section which the processor which performs a contents inverter has, and the contents of description of the <input> tag which is the component object 4101 described by the 2nd line are memorized similarly.
[0183] Next, the <action> tag which is the action object 4103 is described by the 3rd line.
[0184] With the <action> tag, it is event. Assignment of a property is surely made. It is shown whether it is that by which the script described between the this <action> tag and the termination tag corresponding to this is performed by assignment of this property to what kind of event generated by which component. Being aimed at the event of "click actuation" generated with the <input> tag of the 2nd line which has connoted the this <action> tag directly is shown by the example of (A).
[0185] Then, <input> of the 2nd line which is the component object 4101 The <action> tag of the 3rd line is registered into a tag as an action object 4103 which this <input> has (drawing 38 (a)).
[0186] Next, it memorizes as a script in which the script ("alert "clicked"!") in which it is described between the termination tags of the <action> tag of the 5th line (namely, the 4th line) is performed in the <action> tag which is the action object 4103 from the <action> tag of the 3rd line.
[0187] Then, it is <input> of the 2nd line to the 6th line. The termination tag corresponding to a tag is described. here, generation (namely, the 8th line of drawing 36 (B)) of the <input> tag begins -- having (drawing 38 (b)) -- <input> The contents of storage about a tag (namely, "inputname= which is the contents of description of the 1st line of drawing 36 (A) -- "field1" -- ") are outputted.
[0188] Then, this <input> The notice of output directions is performed to the <action> tag which is

the action object 4103 previously registered into the tag (drawing 38 (b)).
[0189] If this notice of an output is received with the <action> tag, the contents of assignment about the property mentioned above about the this <action> tag will be sent to the component object 4101, and will be added to the <input> tag under generation (drawing 38 (d)). At the example of drawing 36 (A), it is event. Since the property is specified as "click", "onclick=" is made to be outputted to the <input> tag. Here, generation of the rereeling reel of the <input> tag by the component object 4101 is completed.
[0190] Furthermore, the <action> tag which is the action object 4103 by which the script mentioned above is held is registered into the <form> tag which is the container object 4102 here (drawing 38 (e)).
[0191] The termination tag corresponding to the <form> tag of the 1st line is described by the last at the 7th line of drawing 36 (A). Here, generation (drawing 38 (f)) of the <form> tag which is the container object 4102 is started, and the 1st-line output of drawing 36 (B) is performed first.
[0192] Then, generation of the call part of a script is directed to the script call section object 4104 (drawing 38 (g)). Here, first, the identifier (action123 (target)) of a processing script is generated automatically, the 2nd line of drawing 36 (B) and the 5th line are outputted, and the 4th line by which the return value of this processing script is defined continuously is outputted. Furthermore, the identifier of a processing script is outputted to the <input> tag of the 8th line which is the component object 4101 which is the call origin of this processing script.
[0193] And finally it is directed to the <action> tag which is the action object 4103 previously registered into the <form> tag (drawing 38 (h)), and the script mentioned above is outputted as the 3rd line of drawing 36 (B). With the <form> tag which is the container object 4102, the 6th line of drawing 36 (B), the 7th line, and the 9th line are outputted, and generation of the HTML sentence of drawing 36 (B) is completed next.
[0194] The things which were explained by the above and which showed contents transform processing with the flow chart are drawing 39 and drawing 40 . Next, this flow chart is explained.
[0195] First, in drawing 39 , initiation of processing reads a descriptive text as shown in drawing 36 (A) from a head line.
[0196] If it is judged whether the action tag is described by this read line (S4201) and it is not an action tag (the judgment result of S4201 is No), the contents of the tag described by this line are memorized by the storage section (S4202), after that, processing will return to S4201 and the judgment processing about the next line of this line will be repeated.
[0197] On the other hand, if the action tag is described by the read line (the judgment result of S4201 is Yes), this action tag will be registered into the component tag containing this action tag (S4203), and the processing script continuously described between the initiation tag of this action tag and the termination tag will be memorized by the storage section (S4204).
[0198] Next, the next line of the line the termination tag of an action tag is described to be is read, and it is judged whether the termination tag of a component tag is described by this line (S4205). Consequently, if the termination tag of a component tag is not described by this line (the judgment result of S4205 is No), processing returns to S4201 and the judgment processing about this line is repeated.
[0199] On the other hand, if the termination tag of a component tag is described by this line at this line (the judgment result or Yes of S4205), the contents of description about this component tag that the generation about this component tag is started and is memorized by the storage section will be outputted (S4206), and the property of the action tag registered into this component tag will be added further (S4207).
[0200] Next, processing progresses to drawing 40 , the contents of the property of the added action tag are investigated here, and it is judged whether the event specified with this property is what is supported by this component (S4208). Consequently, if the event specified is not supported by this component (the judgment result of S4208 is No), the error notification which shows that an error exists in the original descriptive text as a processing result of this contents transform processing will be outputted (S4209), and processing will be completed.
[0201] On the other hand, if the event specified is supported by this component (the judgment result of S4208 is Yes), generation of a component tag will be completed (S4210) and the generated

component tag will be temporarily kept by the storage section. Furthermore, the action tag registered into this component tag is registered into a container tag.
[0202] Next, the next line of the line the termination tag of a component tag was described to be is read, and it is judged whether the termination tag of a container tag is described by this line (S4212). Consequently, if the termination tag of a container tag is not described by this line (the judgment result of S4212 is No), processing returns to S4201 (drawing 39), and the judgment processing about this line is repeated.
[0203] On the other hand, if the termination tag of a container tag is described by this line at this line (the judgment result or Yes of S4212), the contents of description about this container tag that the output about this container tag is started and is memorized by the storage section will be outputted (S4213).
[0204] Then, the identifier of a processing script is generated automatically by the script call section object, and it is outputted (S4214), and when the identifier is described by the component tag currently kept still more temporarily, a component tag and a processing script are associated.
[0205] Then, the processing script which relate with the action tag registered into this container tag, it is described, and the storage section was made to memorize previously is outputted (S4215), and finally the output of the <script> tag, the output of the termination tag of a container tag, etc. are performed, and generation of a container tag is completed (S4216).
[0206] By performing the above processing, a HTML sentence as shown in drawing 36 (B) from a descriptive text as shown in drawing 36 (A) is generated.
[0207] In addition, although you may make it make this contents transform processing process with the processing engine independently prepared in a server, it is combined with display processing mentioned above, and you may make it make it process with the framework engine explained in the 3rd example of this invention mentioned above. At this time, the JSP source which is a candidate for conversion in contents transform processing is described as an element of the renderer in the view with which the definition of the appearance of the screen for displaying a model is performed.
[0208] Next, application of this 4th example is explained.
[0209] Although the script which defines processing by drawing 36 was described to description (the 4th line) at the descriptive text (A), the technique using the processing script components-ized is explained first, without describing this processing script in a descriptive text (A).
[0210] Drawing 41 is drawing showing the example of script description in the case of preparing a processing script independently.
[0211] the descriptive text shown in drawing 36 (A) -- all tags -- sf: the descriptive text shown in drawing 41 (A) although the prefix was attached -- an action tag <action> -- my -- : -- the prefix is attached. This tag <my:action> It is called the mire cushion tag, and when this tag is detected in contents transform processing, the processing script currently prepared beforehand is made to be outputted by processing of S4215 (drawing 40). In addition, sf explained to this until now: Especially the action tag that attached the prefix is called the custom-made action tag.
[0212] The example of a definition of the processing script which checks the number of the minimum alphabetic characters of a character string which is a script corresponding to the mire cushion tag described by drawing 41 (A) is shown in drawing 42 . MinLength specified as a mire cushion tag in the example of a definition shown in drawing 42 setMinLength() for receiving a property is defined and making the processing script which embedded further the contents of the property mentioned above to the output object Writer in the outputFunctionBody method output is directed.
[0213] It is Variable min, when a processing script as shown in drawing 42 in processing of drawing 39 of S4204 will be memorized by the storage section and the processing script will be outputted in processing of subsequent drawing 40 of S4215, if contents transform processing shown in drawing 39 and drawing 40 is performed. The output with which substitution of a value was made is performed. Consequently, the HTML sentence shown in drawing 41 (B) from the descriptive text of drawing 41 (A) is generated.
[0214] Next, drawing 43 is explained. This drawing is drawing showing the example of script description in case two or more actions correspond to the same event.
[0215] <input> shown in the 2nd line by the descriptive text of drawing 43 (A) The tag has two

action tags, the 3rd line and the 6th line. In such a case, in contents transform processing shown in drawing 39 and drawing 40 , according to an operation of judgment processing of S4205 of drawing 39 , processing which lasts to S4205 from S4201 is performed twice, consequently these two action tags are registered into the <input> tag, and the processing script about each action tag is memorized further.

[0216] Then, it sets to processing of drawing 39 of S4207, and is <input> from two action tags. "event" specified as these two action tags in contents transform processing if a property is added to a tag It is recognized that the contents are the same.

[0217] Based on this recognition result, automatic generation of the processing script for performing in order two processing scripts shown in the 8th line (part of (a)) from the 2nd line of drawing 43 (b) in addition to automatic generation of the identifier of a processing script is performed by contents transform processing by the script call section object in processing of drawing 40 of S4214. The correspondence relation between the same event and two or more processing scripts is established by automatic generation of the processing script by this script call section object.

[0218] Then, in S4215, the 12th line (part of (b)) and 13th line to 19th-line (part of (c)) output is performed from the 9th line of drawing 43 (B).

[0219] Next, drawing 44 is explained. This drawing shows the example of script description in case an event occurs with a container tag.

[0220] In drawing 44 (A), it sets to description of the action tag of the 6th line, and is event. The property is specified as ""form.submit"." It is shown that this description is an action tag about the container tag with which the mame property is specified as "form".

[0221] About the container tag with which assignment of such a property is made, registration of the action tag which is processing of drawing 39 of S4203 is made to be performed instead of a component tag to a container tag in contents transform processing shown in drawing 39 and drawing 40 . Furthermore, the processing performed to the component tag which lasts to S4210 from S4207 in the case of the output of the contents of description about the container tag in processing of drawing 40 of S4213 is made to be performed to a container tag. By carrying out like this, the 14th line (line of (a)) of drawing 44 (B) comes to be generated, and it becomes possible to perform a script to the event generated with a container tag.

[0222] The configuration of the each processing engine of a server and client which are used by the system which carries out each example of this invention explained to the above is shown in drawing 45 . Each of these has CPU5001, the storage section 5002, the input section 5003, the output section 5004, and the I/F section 5005, and each component is mutually connected through the bus 5006.

[0223] If the function of each component is explained, CPU (central processing unit)5001 will control each component by performing a control program.

[0224] The storage section 5002 is equipped with ROM (read-only memory), RAM (random access memory), magnetic storage, etc., and is used as the work area or the various data storage fields at the time of the storage of a control program which makes CPU5001 control each component, and CPU5001 performing a control program.

[0225] The input section 5003 has the mouse, the keyboard, etc. and various kinds of data corresponding to actuation by the user are acquired.

[0226] The output section 5004 has the display etc., presents various kinds of data and notifies a user of them.

[0227] The I/F section 5005 offers the interface function for connecting with a network, and enables data transfer with other devices through a network.

[0228] In addition, the configuration shown in this drawing 45 is the same configuration as what the standard computer has, therefore, of course, can also carry out this invention by standard computer.

[0229] Drawing 46 is drawing explaining the offer approaches, such as various kinds of software programs executed with information processors, such as a computer concerning this invention. A program etc. is offered by the approach of the arbitration in the three approaches of for example, the following (a) - (c).

[0230] (a) It is installed in the information processors 5301, such as a computer, and is provided. In this case, a program etc. is pre-installed for example, before shipment.

[0231] (b) It is stored in the portable mold record medium 5302, and is provided. In this case, the

program stored in the portable mold storage 5302 is installed in the external storage of the information processors 2301, such as a computer. As an example of the portable mold storage 5302, there are a floppy (trademark) disk, CD-ROM, a magneto-optic disk, a DVD-ROM, etc.
[0232] (c) It is provided from the program offer server 5304 on a network 5303. In this case, that program etc. is acquired when the information processors 5301, such as a computer, download fundamentally the program stored in the program offer server 5304. In this case, activation of the software program concerned is attained by transmitting the transmission signal which modulates a subcarrier and is obtained through the network 5303 which is a transmission medium from the program offer server 5304 with the data signal expressing a software program, restoring to the transmission signal which received in the information processor 5301, and reproducing a software program.
[0233] (Additional remark 1) Web system characterized by being the Web system which returns a response to a client, changing said request into a data object, processing [to process the request from a client by the server,] it by said server, and changing and returning the data object which it is as a result of processing to the response to said client.
[0234] (Additional remark 2) A contents conversion means of an input to be a system for developing and performing Web application, and to change the contents of an input into a data object, Processing logic equipped with two or more manipulation routines, and the class of said data object and the 1st external declaration file which maps the combination of a command in said each manipulation routine, The Web application development and the executive system characterized by having a manipulation-routine decision means to determine a suitable manipulation routine from the manipulation routine with which said processing logic is equipped based on the class of said data object, said command, and said 1st external declaration file.
[0235] (Additional remark 3) They be the Web application development and the executive system which be the Web application development and an executive system of additional remark 2 publication , and be characterize by for said contents conversion means of an input make the specific item of the data input page described in HTML the class name of said data object , make the identifier of each input column prepare in this page for data inputs correspond to the attribute of said data object , and generate the program for data objects automatically .
[0236] (Additional remark 4) The Web application development and executive system characterized by having the processing logic which is a system for developing and performing Web application, and is equipped with two or more manipulation routines, and the processing result of said processing logic and the 2nd external declaration file which maps the combination of the class of data object in the component for a display.
[0237] (Additional remark 5) The Web application development and executive system characterized by having the template file which specified the method of arrangement of two or more components for a display which are the Web application development and the executive system of additional remark 4 publication, and are further arranged to the page to display, and outputting the processing result of two or more of said processing logic based on said template file.
[0238] (Additional remark 6) The XML mapping file which are the Web application development and the executive system of additional remark 2 publication, and maps the tag and data object of XML further, XML which performs the interconversion of the tag of XML, and a data object Data It has a Binding engine. It is said XML when the tag described by XML as said contents of an input is received. Data A Binding engine Said XML which received is changed into said data object based on the tag and said XML mapping file of said XML which received. Said manipulation-routine decision means Based on said data object, the tag of said XML which received, and said 1st external declaration file, a suitable manipulation routine is determined from the manipulation routine in said processing logic. Said XML Data A Binding engine is the Web application development and an executive system characterized by what the data object obtained as a processing result of said processing logic is changed and outputted for to the tag of XML.
[0239] (Additional remark 7) They are the Web application development and the executive system which are the Web application development and an executive system of additional remark 6 publication, and is characterized by said XML mapping file mapping the tag of XML which performs the same processing as the data based on a certain HTTP in the data object of the same

class as the data based on said a certain HTTP.
[0240] (Additional remark 8) They are the Web application development and the executive system which are the Web application development and an executive system of additional remark 2 publication, and is characterize by for each manipulation routine in said processing logic branch on a scope small from a bigger scope when the attribute definition is carry out in order with a big time scope at said scope of a system, application, a session, or a four steps of inside request and said processing logic processes, and advance processing.
[0241] (Additional remark 9) The Web application development and executive system which are the Web application development and an executive system of additional remark 8 publication, and is characterized by preparing the interface of pretreatment or after treatment in said manipulation routine by which the attribute definition of said scope of any or ** is carried out.
[0242] (Additional remark 10) The Web application development and executive system which are the Web application development and an executive system of additional remark 9 publication, and is characterized by preparing the check routine for controlling a condition to the interface of said pretreatment.
[0243] (Additional remark 11) The Web application development and executive system which are the Web application development and an executive system of additional remark 9 publication, and is characterized by preparing the error handling processing for making the interface of said after treatment recover the error for said every manipulation routine by which the attribute definition of said scope is carried out.
[0244] (Additional remark 12) The Web application development and executive system which are the Web application development and an executive system of additional remark 8 publication, and is characterized by preparing a single thread actuation limit in either of said manipulation routines by which the attribute definition of said scope is carried out.
[0245] (Additional remark 13) The step which is the record medium which recorded the program read by it when used by computer, and changes the contents of an input into a data object, The external declaration file which maps the combination of the class of said data object, a command, and the class of said data object and a command in each manipulation routine, The record medium which recorded the program for making the step which is alike, is based and determines a suitable manipulation routine from the manipulation routine in processing logic perform to said computer and which can be computer read.
[0246] (Additional remark 14) The request from a client is processed by the server. Each object which returns a response to a client and which is a Web system and is processed according to said request by said server The attribute definition about a time scope based on the relative physical relationship on a time-axis in case each is processed is carried out. Said server The Web system characterized by branching to the object by which the small time scope is defined, and advancing processing of the object according to said request from the object by which the bigger time scope is defined.
[0247] (Additional remark 15) Web system which is a Web system of additional remark 14 publication, and is characterized by making the attribute definition of the scope of a system, application, a session, or the requests at each object at order with a big time scope.
[0248] (Additional remark 16) Web system characterized by performing the object which the request from a client is processed by the server, it is [object] the Web system which returns a response to a client, and actuation of processing by this manipulation routine is supervised [object] at the time of the manipulation-routine call by the server, and makes advance of this processing continue by this server.
[0249] (Additional remark 17) It is prepared in the server side which delivers and receives various kinds of data between clients. It is Web application generation equipment which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed. A data object storing means by which the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data is stored, A definition statement storing means by which the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML is stored, By associating the data which this data object has based on said interface mounted

in said data object, and said definition statement Web application generation equipment characterized by having a HTML sentence generation means to generate the HTML sentence expressing the Web page screen where these data are displayed.
[0250] (Additional remark 18) It is Web application generation equipment of additional remark 17 publication, and said definition statement is JavaServer. It is Web application generation equipment which is defined by Pages (JSP) and characterized by what said HTML sentence generation means generates said HTML sentence for from the definition statement defined by JSP.
[0251] (Additional remark 19) It is Web application generation equipment which has further a data class acquisition means to acquire the data class which is Web application generation equipment of additional remark 17 publication, and is the attribute defined corresponding to said DS, and is characterized by what said HTML sentence generation means chooses for said definition statement related with the data which said data object has based on said data class.
[0252] (Additional remark 20) It is Web application generation equipment of additional remark 17 publication. A request acquisition means to acquire this request that is a request containing the input data to the entry form shown in the Web page screen expressed by the HTML sentence generated by said HTML sentence generation means, and is sent from said client, It is the character string contained in the HTML sentence which is contained in said request with said data, and which was generated by said HTML sentence generation means. It is based on this character string that consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface used as the foundation of generation of this HTML sentence, and this interface. Web application generation equipment characterized by having further a renewal means of data to update the data of the specific location in the DS expressed by the interface specified by this character string to the data contained in this request.
[0253] (Additional remark 21) The character string which is Web application generation equipment of additional remark 20 publication, and is contained in said request with said data is Web application generation equipment characterized by showing the parameter name which refers to the data inputted to this entry form.
[0254] (Additional remark 22) Web application generation equipment which is Web application generation equipment of additional remark 20 publication, and is characterized by containing said data and said character string about the HTML sentence generated by said request based on a different interface.
[0255] (Additional remark 23) It is Web application generation equipment which is Web application generation equipment of additional remark 20 publication, and is characterized by said renewal means of data updating said data which the data object stored in said data object storing means has to the data contained in said request.
[0256] (Additional remark 24) The character string which is Web application generation equipment of additional remark 20 publication, and is contained in said request with said data To the character string which consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface used as the foundation of generation of said HTML sentence, and this interface Furthermore, it changes combining the character string which consists of a character string which shows the specific location in the DS expressed by the character string which specifies the interface showing the DS which the data constellation of this location has, and this interface. Said renewal means of data is Web application generation equipment characterized by what the data of the specific location in the DS which it has further in the specific location in the DS specified by said character string are updated for to the data contained in said request.
[0257] (Additional remark 25) By the server side which delivers and receives various kinds of data between clients It is the approach of generating the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed. The data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data is acquired. The definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML is acquired. By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The Web application generation method characterized by having ** which generates the HTML sentence expressing the Web page screen where these data are displayed.

[0258] (Additional remark 26) By the server side which delivers and receives various kinds of data between clients It is the record medium which recorded the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by performing a computer are displayed perform to this computer. The control which acquires the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data, The control which acquires the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The storage which memorized the control program which makes the control which generates the HTML sentence expressing the Web page screen where these data are displayed perform to a computer.
[0259] (Additional remark 27) By the server side which delivers and receives various kinds of data between clients It is the computer data signal embodied by the subcarrier containing the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by performing a computer are displayed perform to this computer. The control which acquires the data object which mounts the interface which this control program has data with which said client is provided, and expresses the DS of these data, The control which acquires the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The control which generates the HTML sentence expressing the Web page screen where these data are displayed is made to perform to a computer.
[0260] (Additional remark 28) It is prepared in the server side which delivers and receives various kinds of data between clients. A processing logic storing means by which the processing logic by which it is Web application generation equipment which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed, and the contents of processing are defined is stored, An execution condition storing means by which the execution condition of said processing logic is stored, and a processing logic name generation means to generate the character string used as the name of said processing logic, It is the HTML sentence which calls said processing logic using said character string. Web application generation equipment characterized by having a HTML sentence generation means to generate this HTML sentence that makes processing in which this processing logic is called and performed when the event corresponding to said execution condition occurs in said client perform to this client.
[0261] (Additional remark 29) It is Web application generation equipment characterized by being Web application generation equipment of additional remark 28 publication, and defining said processing logic in the script.
[0262] (Additional remark 30) Web application generation equipment which is Web application generation equipment of additional remark 28 publication, and is characterized by storing in both said processing logic storing means and said execution condition storing means this processing logic and this execution condition in the same component with which said processing logic and said execution condition are defined, respectively.
[0263] (Additional remark 31) It is Web application generation equipment of additional remark 28 publication. Said two or more processing logic is stored in said processing logic storing means. When either of said each execution condition of two or more of said processing logic is the same It has further a processing logic generation means by which this execution condition generates the processing logic which performs this same processing logic in order. Said character string generation means The character string used as a respectively different name to said two or more processing logic and the processing logic generated by said processing logic generation means is generated. Said HTML sentence generation means It is the HTML sentence which calls the processing logic generated by said processing logic generation means using said character string. Web application generation equipment characterized by what this HTML sentence that makes processing in which this processing logic is called and performed when the event corresponding to said same execution condition occurs in said client perform to this client is generated for.
[0264] (Additional remark 32) Web application generation equipment characterized by having

further a check means to be Web application generation equipment of additional remark 28 publication, and to check validity correspondence-related [with the execution condition of said processing logic and this processing logic].

[0265] (Additional remark 33) By the server side which delivers and receives various kinds of data between clients It is the Web application generation method which generates the HTML sentence expressing the Web page screen where the data with which this client is provided are displayed. The character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined is generated. It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The Web application generation method characterized by what this HTML sentence that makes the processing to say perform to this client is generated for.

[0266] (Additional remark 34) By the server side which delivers and receives various kinds of data between clients It is the record medium which recorded the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by performing a computer are displayed perform to this computer. The control which generates the character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined, It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The storage which memorized the control program which makes the control which generates this HTML sentence that makes the processing to say perform to this client perform to a computer.

[0267] (Additional remark 35) By the server side which delivers and receives various kinds of data between clients It is the computer data signal embodied by the subcarrier containing the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by performing a computer are displayed perform to this computer. The control which generates the character string used as the name of this processing logic that this control program is processing logic by which the contents of processing are defined, and is prepared beforehand, It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The control which generates this HTML sentence that makes the processing to say perform to this client is made to perform to a computer.

[0268] (Additional remark 36) Program for performing the step which determines a suitable manipulation routine as a computer from the manipulation routine in processing logic based on the external declaration file which maps the combination of the step which changes the contents of an input into a data object, the class of said data object, a command, and the class of said data object and a command in each manipulation routine.

[0269] (Additional remark 37) By the server side which delivers and receives various kinds of data between clients It is the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by performing a computer are displayed perform to this computer. The control which acquires the data object which mounts the interface which has data with which said client is provided, and expresses the DS of these data, The control which acquires the definition statement which defines the method of presentation in said Web page screen about the data which said data object has by description by HTML, By associating the data which this data object has based on said interface mounted in said data object, and said definition statement The control program for making the control which generates the HTML sentence expressing the Web page screen where these data are displayed perform to a computer.

[0270] (Additional remark 38) By the server side which delivers and receives various kinds of data between clients It is the control program which makes the control which generates the HTML sentence expressing the Web page screen where the data with which this client is provided by

performing a computer are displayed perform to this computer. The control which generates the character string used as the name of this processing logic that is processing logic and is prepared beforehand with which the contents of processing are defined, It is the HTML sentence which calls said processing logic using said character string. When the event corresponding to this execution condition that is an execution condition about said processing logic, and is prepared beforehand occurs in said client, call and perform this processing logic. The control program for making the control which generates this HTML sentence that makes the processing to say perform to this client perform to a computer.

[0271]

[0272]

[Effect of the Invention] As explained to the detail above, according to this invention, cooperation of the logic which specifies HTML which specifies a screen display in Web application system, a data object, and the contents of processing will serve as a non-dense, can raise the reusability of each module, and can raise development effectiveness and maintainability. Moreover, by branching the processing in a server to a small thing from what has a big time scope, and advancing processing, a thread safe can be realized, or pretreatment and after treatment can be added to the processing in each scope, and the effectiveness that pre-decision of processing and error processing can be performed flexibly can be acquired.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] It is drawing showing the outline of this invention.
[Drawing 2] It is drawing showing the principle configuration of this invention.
[Drawing 3] It is drawing showing the system configuration of the 1st example of this invention.
[Drawing 4] It is drawing showing the situation of concrete actuation of the system of the 1st example of this invention.
[Drawing 5] It is drawing which explains the actuation outline to be the system configuration of the application shown in drawing 4 (until it receives and processes the request from a client).
[Drawing 6] It is drawing which explains the actuation outline to be the system configuration of the application shown in drawing 4 (until it returns a response to a client).
[Drawing 7] It is drawing in which (**) and (b) show a part of command mapping, and (c) shows a part of page mapping.
[Drawing 8] It is Java for an input about the request data from a client. It is drawing showing the programme description changed into Bean.
[Drawing 9] It is drawing showing the programme description which opts for processing of the logic which should be performed through command mapping.
[Drawing 10] User logic is drawing showing the programme description which sets up the data object for a display.
[Drawing 11] It is drawing showing the programme description which determines the page which should be displayed through page mapping.
[Drawing 12] It is drawing showing the programme description which outputs the display screen from JSP.
[Drawing 13] It is drawing explaining the template in the case of arranging two or more display articles to the display screen.
[Drawing 14] It is drawing (the 1) showing the system configuration of the 2nd example of this invention.
[Drawing 15] It is drawing (the 2) showing the system configuration of the 2nd example of this invention.
[Drawing 16] It is drawing showing the outline of the system which processes an XML file of operation (until it receives and processes an XML file).
[Drawing 17] It is drawing showing the outline of the system which processes an XML file of operation (when an XML file is received, until it returns a processing result).
[Drawing 18] (**) is drawing explaining the scope of the object which constitutes the system of this invention, and (b) is drawing showing the correlation between objects.
[Drawing 19] (**) is drawing explaining the request processing from a client, and (b) is drawing explaining mounting of SingleThreadModel.
[Drawing 20] (**) is SingleThreadModel. It is drawing showing concrete mounting, and (b) is drawing showing the example of programme description of mounting of SingleThreadModel.
[Drawing 21] It is a sequence diagram in the case of processing the request of a client.
[Drawing 22] (**) is drawing explaining the structure which calls back a browser from the handler of logic in a general application server, and (b) is drawing showing the example of programme description.

[Drawing 23] (**) is drawing explaining the structure which requests to a browser from the handler of logic in the application server of this invention, and (b) and (c) are drawings showing the example of programme description.
[Drawing 24] It is drawing showing the system configuration of the 3rd example of this invention.
[Drawing 25] It is drawing showing the example of declaration of a table model interface.
[Drawing 26] In the 3rd example of this invention, it is drawing explaining signs that a HTML sentence is generated.
[Drawing 27] It is the flow chart (the 1) which shows the contents of processing of the control processing at the time of a display.
[Drawing 28] It is the flow chart (the 2) which shows the contents of processing of the control processing at the time of a display.
[Drawing 29] It is the flow chart (the 3) which shows the contents of processing of the control processing at the time of a display.
[Drawing 30] It is drawing explaining the renderer element tag of <name>.
[Drawing 31] It is drawing showing the example of a definition of the renderer which used the renderer element tag of <name>.
[Drawing 32] It is the flow chart which shows the contents of processing of storage region management processing.
[Drawing 33] It is drawing explaining the outline of the actuation at the time of the request in the system shown in drawing 24 .
[Drawing 34] It is the flow chart which shows the contents of processing of the control processing at the time of a request.
[Drawing 35] It is drawing showing the example of the client/server system which carries out the 3rd example of this invention.
[Drawing 36] It is drawing showing signs that a HTML sentence is generated by the 4th example of this invention.
[Drawing 37] It is drawing showing the configuration of the object used in the 4th example of this invention.
[Drawing 38] It is drawing explaining the outline of contents transform processing.
[Drawing 39] It is the flow chart (the 1) which shows the contents of processing of contents transform processing.
[Drawing 40] It is the flow chart (the 2) which shows the contents of processing of contents transform processing.
[Drawing 41] It is drawing showing the example of script description in the case of preparing a processing script independently.
[Drawing 42] It is drawing showing the example of a definition of the processing script which checks the number of the minimum alphabetic characters of a character string.
[Drawing 43] It is drawing showing the example of script description in case two or more actions correspond to the same event.
[Drawing 44] It is drawing showing the example of script description in case an event occurs with a container tag.
[Drawing 45] It is drawing showing the configuration of the each processing engine of a server and client which are used for the system which carries out each example of this invention.
[Drawing 46] It is drawing explaining the offer approaches, such as a software program concerning this invention.
[Drawing 47] It is drawing showing the operation gestalt of a business application.
[Drawing 48] It is drawing showing the system configuration of the application server using Servlet.
[Drawing 49] It is drawing showing the system configuration of the application server using JSP.
[Description of Notations]
101 HTML
102 Screen Data
103 Logic
201 JSP for Input
202 JSP for Display

203 Java for Input Bean
204 Java for Display Bean
205 User Logic
206 UJI Engine
207 Mapping File
301 HTTP Request
302 Front Component
303 Java for Input Bean
304 UJI Engine
305 Command Mapping
306 Page Mapping
307 User Logic
308 Template
309 Java for Display Bean
310 JSP for Display
401 User List Display Screen
402 User Condition Edit Display
403 Profile Edit Display
404 User Registration Screen
405 Log in Carbon Button
406 User Registration Carbon Button
407 Modification Carbon Button
408 Log Out Carbon Button
409 Profile Edit Carbon Button
410 Modification Carbon Button
411 Returning Carbon Button
412 Modification Carbon Button
413 Returning Carbon Button
501 User List Display Screen
502 User Condition Edit Display
503 Profile Edit Display
504 User Registration Screen
505 Java Bean (Automatic Generation)
506 UJI
507 Command Mapping
508 Log in Processing
509 Log Out Processing
510 User Status-Change Processing
511 Profile Modification Processing
512 User Registration Processing
601 Java Bean (it Creates within Processing)
602 Page Mapping
801 HTML Page for Data Inputs
802 Java for Input Bean
901 HTML Page for Data Inputs
902 Command Mapping
903 User Logic (Handler)
1001 User Logic (Handler)
1002 Database
1003 Java for Display Bean
1101 Java for Display Bean
1102 Page Mapping
1103 login-succeded.jsp
1104 login-failed.jsp

1201 JSP for Display
1202 Java for Display Bean
1203 Output HTML
1301 User Logic (Handler)
1302 Java for Display Bean (for Banners)
1303 Java for Display Bean (for Menus)
1304 Java for Display Bean (for Contents)
1305 Template
1306 HTML for Output
1401 XML Instance
1402 Command Mapping
1403 UJI Engine
1404 XML Data Binding Engine
1405 Bean Instance
1501 Logic (Handler)
1502 Data Object for Transmission
1503 Transmitting XML Instance
1601 Order Cut-form XML
1602 Shipping Ticket XML
1603 UJI
1604 XML Data Binding Engine
1605 Java Bean
1606 XML Mapfile
1607 Command Mapping
1608 Order Processing
1609 Shipment Processing
1610 Processing
1701 Java Bean
1702 Transmission XML
1801 System
1802 Application
1803 Session
1804 Request
1805 Handler
1901 Front Component
1902 Dispatcher
1903 Application
1904 Session
1905 Handler
1906 Page for Display
1907 Dispatch Context
1908 ResponseBean
1909 Command Map
1910 Page Map
1911 RequestBean
1912 SingleThreadModel
2001 System
2002 Application
2003 Session
2004 Handler
2005 Log in Processing Method
2006 Log Out Processing Method
2007 User Modification Method
2008 Profile Edit Method

2009 User Registration Method
3001 Model Object
3001a The model object for a display
3001b The model object for an input
3002 Model Interface
3003 Framework Engine
3004 JSP for Display
3005 Browser
3006 Front Component
3010 Server
3011 Model Framework Processing Section
3012 Web Server Section
3013 Back-end
3014 Database
3020a, 3020b, 3020c Client
4001 4101 Component object
4002 4102 Container object
4003 4103 Action object
4004 4104 Script call section
4011 ValidInputTag Object
4012 ValidFormTag Object
4013a CustomActionTag Object
4013b MyActionTag Object
5001 CPU
5002 Storage Section
5003 Input Section
5004 Output Section
5005 I/F Section
5006 Bus
5301 Information Processor (Computer)
5302 Portable Mold Record Medium
5303 Network
5304 Program Offer Server
5401 Application Server
5402 Database Server
5403 Client
5501 HTML
5502 Screen Data
5503 Logic
5601 HTML
5602 Screen Data
5603 Data

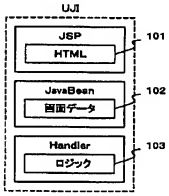
[Translation done.]

* NOTICES *

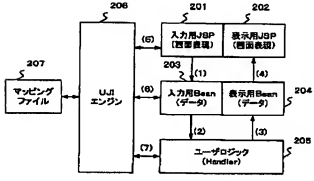
1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DRAWINGS

[Drawing 1]
本発明の概略を示す図

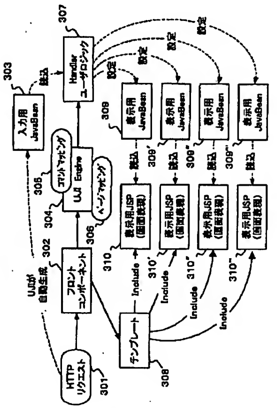


[Drawing 2]
本発明の原理構成を示す図



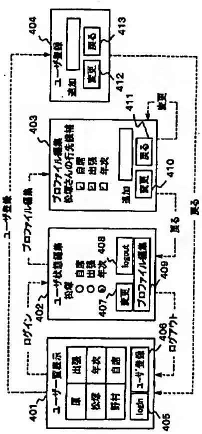
[Drawing 3]

本発明の第1の実施例のシステム構成を示す図



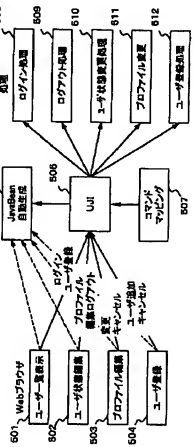
[Drawing 4]

本発明の第1の実施例のシステムの
具体的な動作の様子を示す図



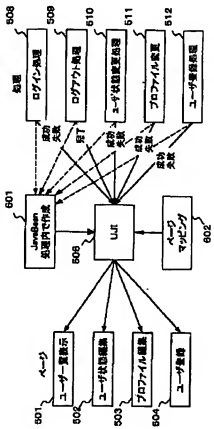
[Drawing 5]

図4に示すアプリケーションのシステム構成とその動作（クライアントからのリクエストを受信し、処理するまで）概要を説明する図

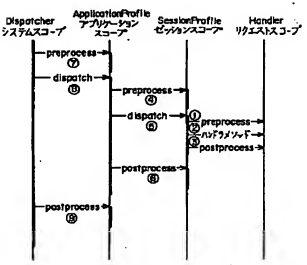


[Drawing 6]

図4に示すアプリケーションのシステム構成とその動作（クライアントからのリクエストを受信し、処理するまで）概要を説明する図



[Drawing 21]
クライアントのリクエストを処理する場合のシーケンス図



[Drawing 7]

(a)、(b)はコマンドマッピングの一部、
(c)はページマッピングの一部を示す図

ユーザ状態編集画面からの分岐

入力データ	コマンド	処理
ユーザ状態	プロファイル編集	プロファイル変更処理
	ログアウト	ログアウト処理

(a)

入力データ	コマンド	処理
ユーザ状態	変更	ユーザ状態変更処理
プロファイル編集		プロファイル変更

(b)

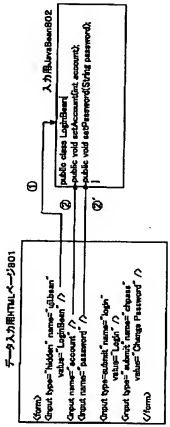
ログイン処理からの分岐

出力データ	処理結果	画面
ユーザ状態	ログイン成功	ユーザ状態編集画面
	ログイン失敗	ログイン失敗画面

(c)

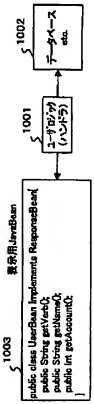
[Drawing 8]

クライアントからのリクエストデータを
入力用Java Beanに変換するプログラム記述を示す図



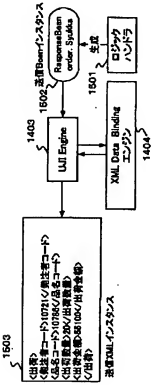
[Drawing 10]

ユーザロジックが表示用データオブジェクト
を設定するプログラム記述を示す図



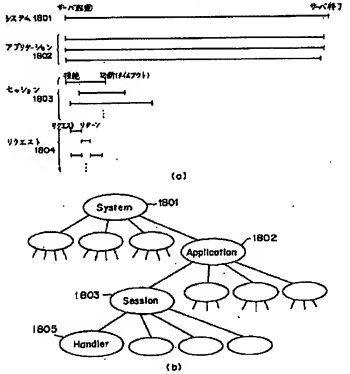
[Drawing 15]

本発明の第2の実施例の
システム構成を示す図(その2)



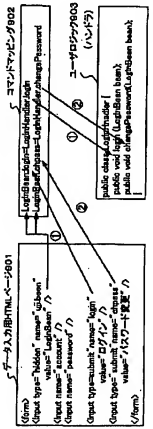
[Drawing 18]

(a)は、本発明のシステムを構成するオブジェクトのスコープを説明する図であり、(b)はオブジェクト間の相関関係を示す図



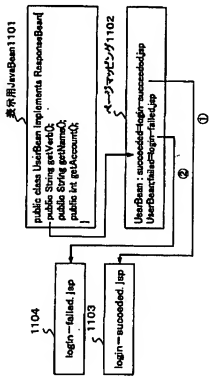
[Drawing 9]

コマンドマッピングを用いて実行すべきロジックの処理を決定するプログラム技術を示す図



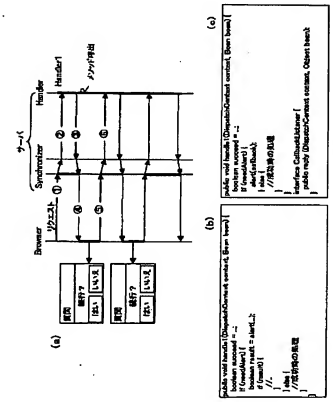
[Drawing 11]

ユーザロジックが表示用データオブジェクト
を設定するプログラム記述を示す図



[Drawing 23]

(a)は、本発明のアプリケーションサーバにおいてロジックハンドラから
ブラウザにリクエストを行う仕組みを説明する図であり、
(b)、(c)はそのプログラム記述例を示す図



[Drawing 25]

テーブルモデルインタフェースの宣言例を示す図

```
public interface Table Model {
  public int getColumnCount();
  public int getRowCount();
  public Object getValueAt(int row ,int col);
  public String getColumnClass(int row ,int col);
  public String getRowClass(int row);
  public void setValueAt(Object value ,int row ,int col);
}
```

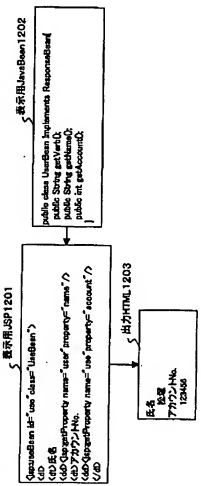
[Drawing 31]

<name>のレンダラエレメントタグを使用したレンダラの定義例を示す図

```
<uomfTableRenderer type="column" clas="editable">
 <td><input name="<uomfname/>"
   value="<uomfvalue/>"/></td>
</uomfTableRenderer>
```

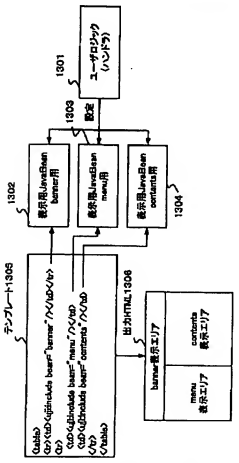
[Drawing 12]

JSPから表示画面を出力するプログラム記述を示す図

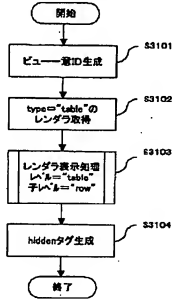


[Drawing 13]

表示画面に複数の表示部品を配置する場合の
テンプレートについて説明する図



[Drawing 27]
表示時の制御処理の処理内容を示すフローチャート
(その1)



[Drawing 30]

<name>のレンダラエレメントタグを説明する図

[ビューの一意ID]. [モデル固有位置]

(A)名前付けの規則

"uji. model. 00001.2_3 "

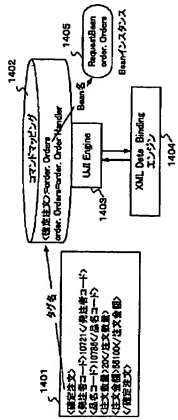
(B)規則に従った名前付けの例

"uji. model. 0000.2_3.000002.1_2 "

(C)ネストしているビューにおける名前付けの例

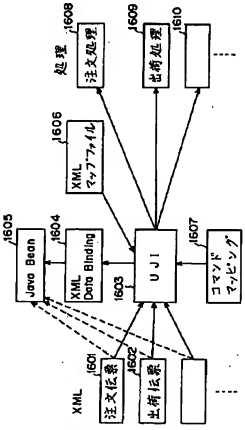
[Drawing 14]

本発明の第2の実施例のシステム構成を示す図(その1)



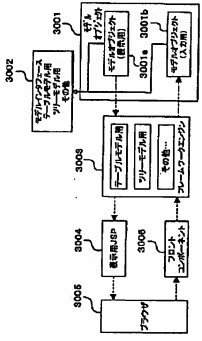
[Drawing 16]

XMLファイルを処理するシステムの動作（XML ファイルを受信して，処理するまで）概要を示す図



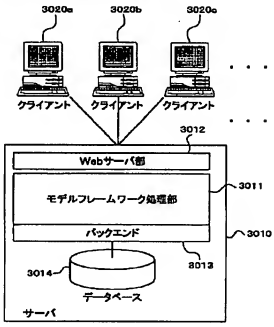
[Drawing 24]

本発明の第3の実施例のシステム構成を示す図



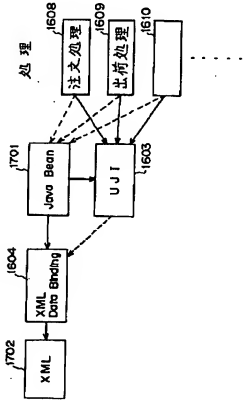
[Drawing 35]

本発明の第3の実施例を実施するクライアント
サーバシステムの例を示す図



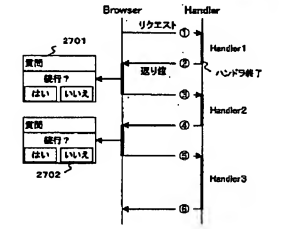
[Drawing 17]

XML ファイルを処理するシステムの動作
(XMLファイルを受信した時に、処理結果を
返すまで) 概要を示す図



[Drawing 22]

(a)は、一般のアプリケーションサーバにおいてロジックハンドラからブラウザにコールバックする仕組みを説明する図であり、(b)はそのプログラム記述例を示す図



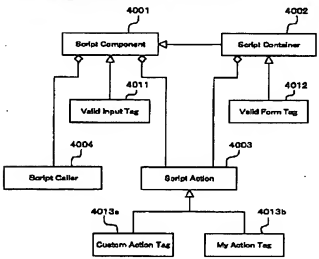
(a)

```
// 本来のイベントハンドラ
public void handle1(DispatchContext context, Bean bean) {
 boolean succeed = ...;
 if (needAlert) {
  //もとの画面用の設定
  // alert用Beanの設定
  return;
 }
 //成功時の処理
 // ...
}
//handle1でアラートを出した返り値のためのイベントハンドラ
public void handle2 (DispatchContext context, AlertBean bean) {
 boolean result = bean. getResult();
 if (result) {
  //...
 }
}
```

(b)

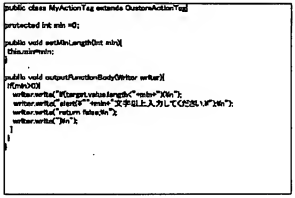
[Drawing 37]

本発明の第4の実施例で使用されるオブジェクトの構成を示す図



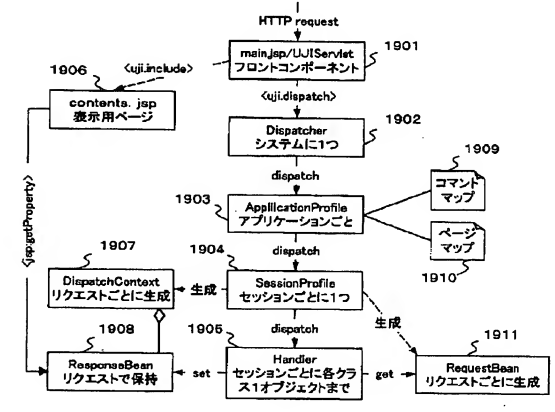
[Drawing 42]

文字列の最小文字数をチェックする
処理スクリプトの定義例を示す図

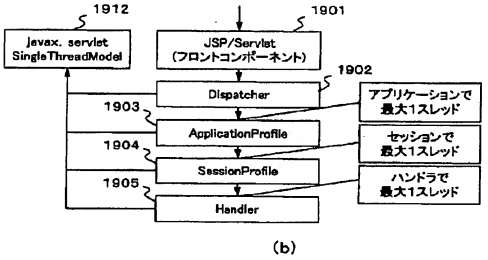


[Drawing 19]

(a)は、クライアントからのリクエスト処理を説明する図
(b)はSingleThreadModel の実装を説明する図



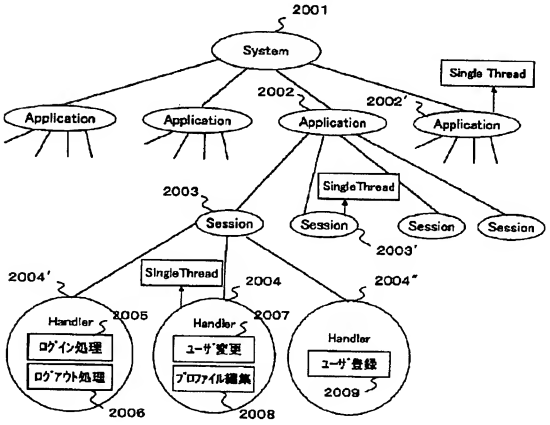
(a)



(b)

[Drawing 20]

(a)は、SingleThreadModel の具体的な実装を示す図
(b)はSingleThreadModel の実装のプログラム記述例を示す図



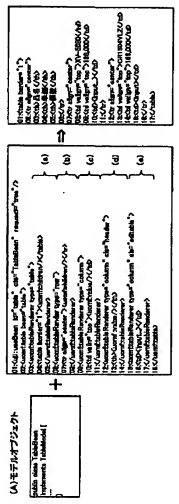
(a)

```
class   MySessionProfile extends SessionProfile implements SingleThread {
        ..............
        ..............
        ..............
}
```

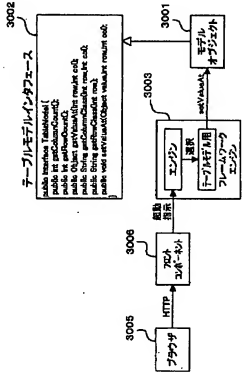
(b)

[Drawing 26]

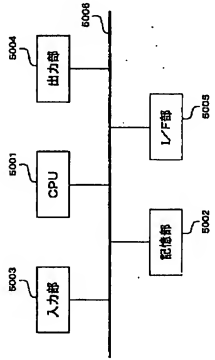
本発明の第3の実施例において、HTML文が生成される様子を説明する図



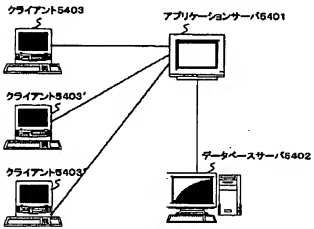
[Drawing 33]
図24に示すシステムにおけるリクエスト時の動作の概要を説明する図



[Drawing 45]
本発明の各実施例を実施するシステムに使用される、サーバの各処理エンジン及びクライアントの構成を示す図

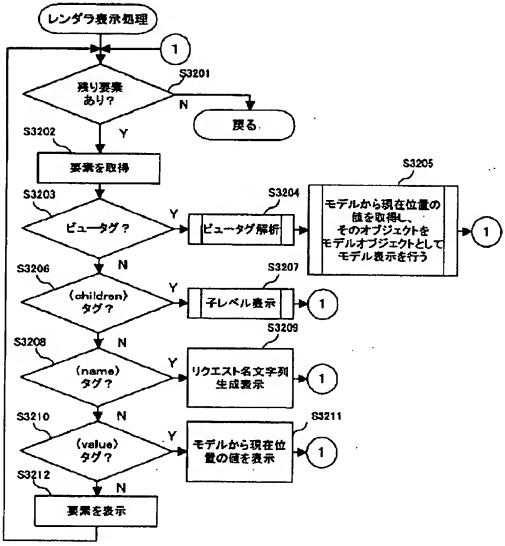


[Drawing 47]
ビジネスアプリケーションの実施形態を示す図

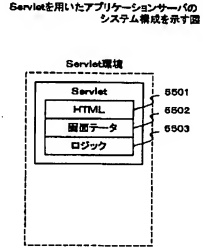


[Drawing 28]

表示時の制御処理の処理内容を示すフローチャート(その2)

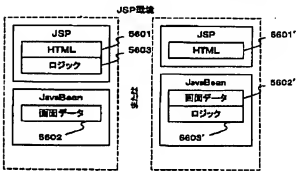


[Drawing 48]



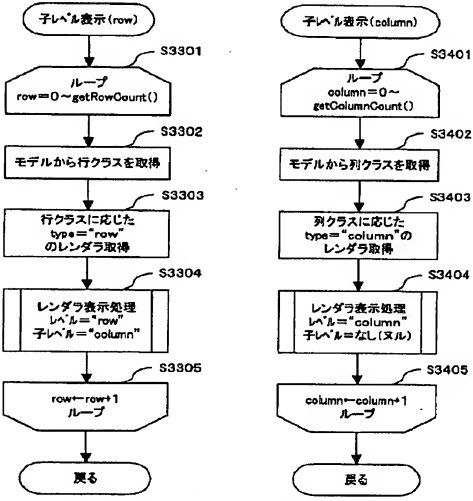
[Drawing 49]

JSPを用いたアプリケーションサーバのシステム構成を示す図



[Drawing 29]

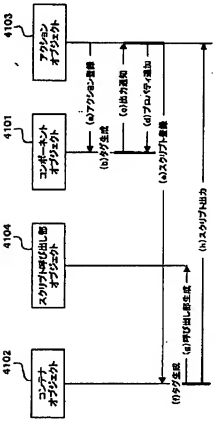
表示時の制御処理の処理内容を示すフローチャート（その3）



(A)子レベルがrowの時の表示処理　(A)子レベルがcolumnの時の表示処理

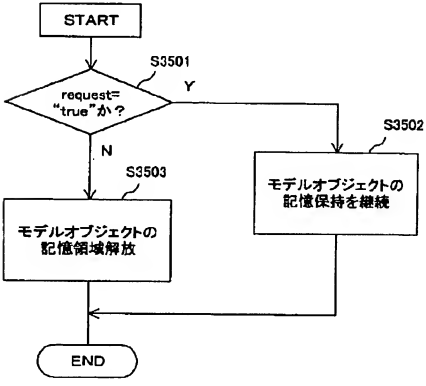
[Drawing 38]

コンテンツ変換処理の概要を説明する図

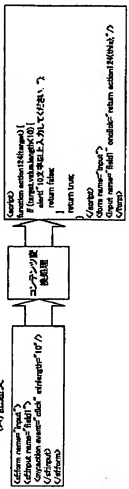


[Drawing 32]

記憶領域管理処理の処理内容を示すフローチャート

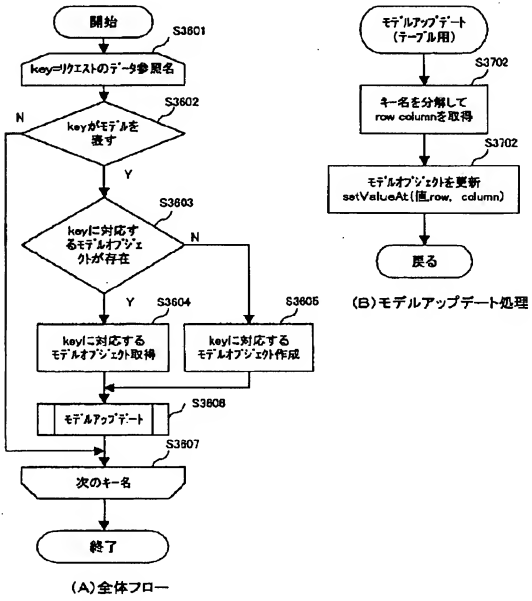


[Drawing 41]
処理スクリプトを別に用意する場合の
スクリプト記述例を示す図



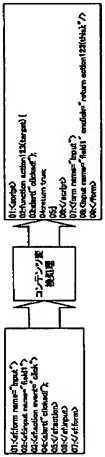
[Drawing 34]

リクエスト時の制御処理の処理内容を示すフローチャート



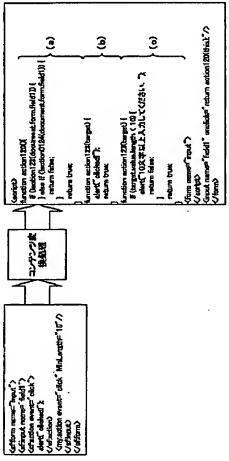
[Drawing 36]

本発明の第4の実施例によってHTML文が生成される様子を示す図



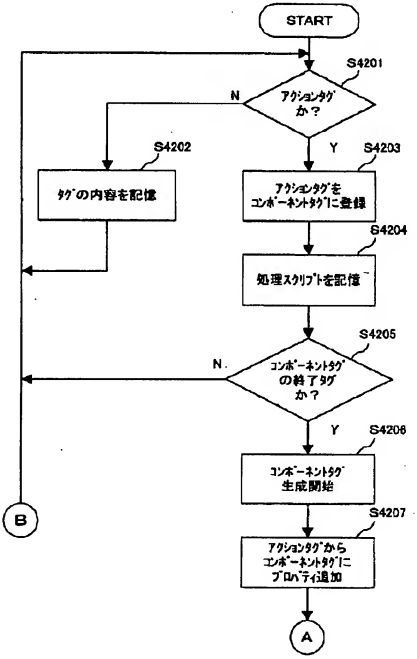
[Drawing 43]

同一のイベントに対して複数のアクションが対応する場合のスクリプト記述例を示す図



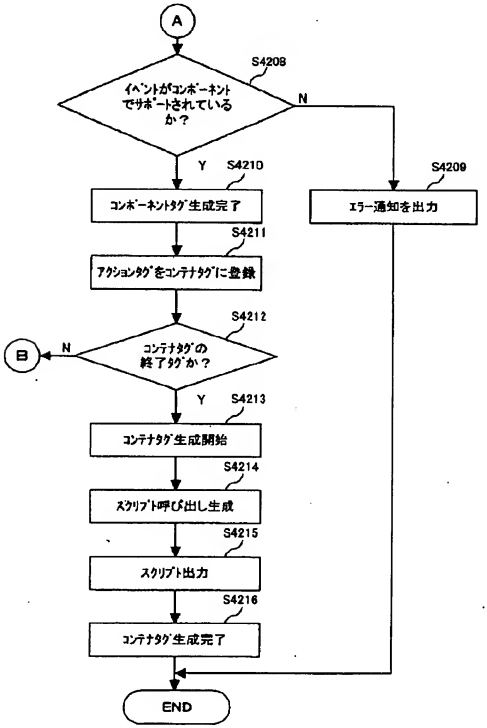
[Drawing 39]

コンテンツ変換処理の処理内容を示すフローチャート
（その1）



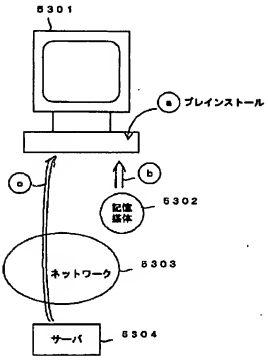
[Drawing 40]

コンテンツ変換処理の処理内容を示すフローチャート
（その2）

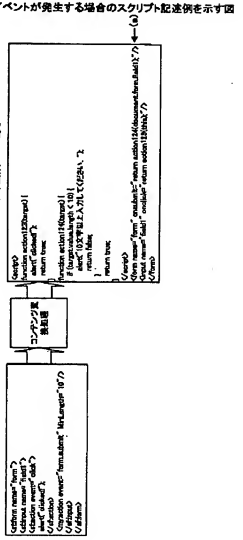


[Drawing 46]

本発明に係わるソフトウェアプログラム等の
提供方法を説明する図



[Drawing 44]
コンテンタグでイベントが発生する場合のスクリプト記述例を示す図



[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－215394
（P2002－215394A）
(43)公開日　平成14年８月２日(2002.8.2)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　9/44		Ｇ０６Ｆ　9/06	６２０Ａ	５Ｂ０７６
			６２０Ｋ	
			６２０Ｃ	

審査請求　未請求　請求項の数23　ＯＬ　（全 46 頁）

(21)出願番号　特願2001－241749(P2001－241749)
(22)出願日　平成13年８月９日(2001.8.9)
(31)優先権主張番号　特願2000－246139(P2000－246139)
(32)優先日　平成12年８月15日(2000.8.15)
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願2000－347977(P2000－347977)
(32)優先日　平成12年11月15日(2000.11.15)
(33)優先権主張国　日本（ＪＰ）

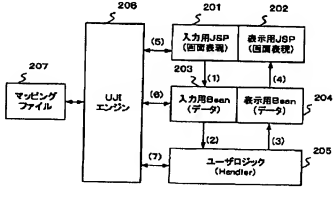
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(72)発明者　松塚　貴英
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　野村　佳秀
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(74)代理人　100074099
弁理士　大菅　義之　（外１名）

Ｆターム(参考)　5B076 DA01 DB01 DB04 DC00 DD05

(54)【発明の名称】　Ｗｅｂアプリケーション開発・実行システム及びＷｅｂアプリケーション生成装置

(57)【要約】
【課題】　Ｗｅｂアプリケーションを実行するアプリケーションサーバを開発する際に、データ、ロジック、画面の各モジュールに分けて記述するフレームワークを提供する。
【解決手段】　Ｗｅｂページの入力内容２０１をデータオブジェクト２０３に変換する入力内容変換手段２０６と、データオブジェクト２０３の種類とコマンドの組合せを各処理ルーチンにマッピングする第１の外部定義ファイル２０７と、処理ルーチンを複数備える処理ロジック２０５と、データオブジェクト２０３の種類とコマンドと第１の外部定義ファイル２０７に基づいて処理ロジック２０５から適当な処理ルーチンを決定する処理ルーチン決定手段２０６と、処理ロジック２０５の処理結果とデータオブジェクト２０４の種類の組合せを表示用コンポーネント２０２にマッピングする第２の外部定義ファイル２０７と、を備える。

本発明の原理構成を示す図



1

【特許請求の範囲】

【請求項１】　クライアントからのリクエストをサーバで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、

前記サーバで、前記リクエストをデータオブジェクトに変換して処理し、処理結果であるデータオブジェクトを前記クライアントへのレスポンスに変換して返すことを特徴とするＷｅｂシステム。

【請求項２】　Ｗｅｂアプリケーションを開発・実行するためのシステムであって、

入力内容をデータオブジェクトに変換する入力内容変換手段と、

複数の処理ルーチンを備える処理ロジックと、

前記データオブジェクトの種類とコマンドの組合せを前記各処理ルーチンにマッピングする第１の外部定義ファイルと、

前記データオブジェクトの種類と前記コマンドと前記第１の外部定義ファイルに基づいて前記処理ロジックの備える処理ルーチンから適当な処理ルーチンを決定する処理ルーチン決定手段と、

を備えることを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項３】　請求項２記載のＷｅｂアプリケーション開発・実行システムであって、

前記入力内容変換手段は、ＨＴＭＬで記述されたデータ入力ページの特定項目を前記データオブジェクトのクラス名とし、該データ入力用ページに設けられている各入力欄の名前を前記データオブジェクトの属性に対応させて、データオブジェクト用プログラムを自動生成することを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項４】　Ｗｅｂアプリケーションを開発・実行するためのシステムであって、

複数の処理ルーチンを備える処理ロジックと、

前記処理ロジックの処理結果とデータオブジェクトの種類の組み合わせを表示用コンポーネントにマッピングする第２の外部定義ファイルと、

を備えることを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項５】　請求項４記載のＷｅｂアプリケーション開発・実行システムであって、

更に、

表示するページに配置する複数の表示用コンポーネントの配置の仕方を規定したテンプレートファイルを備え、

前記テンプレートファイルに基づいて複数の前記処理ロジックの処理結果を出力することを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項６】　請求項２記載のＷｅｂアプリケーション開発・実行システムであって、

更に、

2

ＸＭＬのタグとデータオブジェクトをマッピングするＸＭＬマッピングファイルと、

ＸＭＬのタグとデータオブジェクトとの相互変換を行うＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンを備え、

前記入力内容としてＸＭＬで記述されたタグを受信した場合に、

前記ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンは、前記受信したＸＭＬのタグと前記ＸＭＬマッピングファイルに基づいて、前記受信したＸＭＬを前記データオブジェクトに変換し、

前記処理ルーチン決定手段は、前記データオブジェクトと前記受信したＸＭＬのタグと前記第１の外部定義ファイルに基づいて前記処理ロジック内の処理ルーチンから適当な処理ルーチンを決定し、

前記ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンは、前記処理ロジックの処理結果として得られるデータオブジェクトをＸＭＬのタグに変換して出力する、

ことを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項７】　請求項６記載のＷｅｂアプリケーション開発・実行システムであって、

前記ＸＭＬマッピングファイルは、或るＨＴＴＰによるデータと同一の処理を施すＸＭＬのタグを、前記或るＨＴＴＰによるデータと同一種類のデータオブジェクトにマッピングすることを特徴とするＷｅｂアプリケーション開発・実行システム。

【請求項８】　コンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、

入力内容をデータオブジェクトに変換するステップと、

前記データオブジェクトの種類と、コマンドと、前記データオブジェクトの種類とコマンドの組合せを各処理ルーチンにマッピングする外部定義ファイルと、に基づいて処理ロジック内の処理ルーチンから適当な処理ルーチンを決定するステップと、

を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【請求項９】　クライアントからのリクエストをサーバで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、

前記サーバで前記リクエストに応じて処理される各オブジェクトは、各々が処理されるときの時間軸上における相対的な位置関係に基づいた時間的スコープについての属性定義がされており、

前記サーバは、より大きな時間的スコープが定義されているオブジェクトから小さな時間的スコープが定義されているオブジェクトに分岐して前記リクエストに応じたオブジェクトの処理を進めていくことを特徴とするＷｅｂシステム。

【請求項１０】　クライアントからのリクエストをサー

バで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、
サーバでの処理ルーチン呼び出し時に該処理ルーチンによる処理の動作を監視し、該処理の進行を継続させるオブジェクトを該サーバで実行することを特徴とするＷｅｂシステム。

【請求項１１】　クライアントとの間で各種のデータを授受するサーバ側に設けられ、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成装置であって、
前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトが格納されるデータオブジェクト格納手段と、
前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文が格納される定義文格納手段と、
前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＨＴＭＬ文生成手段と、
を有することを特徴とするＷｅｂアプリケーション生成装置。

【請求項１２】　請求項１１記載のＷｅｂアプリケーション生成装置であって、
前記データ構造に対応して定義される属性であるデータクラスを取得するデータクラス取得手段を更に有し、
前記ＨＴＭＬ文生成手段は、前記データクラスに基づいて前記データオブジェクトの有するデータに関連付ける前記定義文を選択する、
ことを特徴とするＷｅｂアプリケーション生成装置。

【請求項１３】　請求項１１記載のＷｅｂアプリケーション生成装置であって、
前記ＨＴＭＬ文生成手段により生成されたＨＴＭＬ文によって表現されるＷｅｂページ画面に示されている入力フォームへの入力データを含むリクエストであって前記クライアントから送付される該リクエストを取得するリクエスト取得手段と、
前記リクエストに前記データと共に含まれている、前記ＨＴＭＬ文生成手段により生成されたＨＴＭＬ文に含まれている文字列であって、該ＨＴＭＬ文の生成の基礎とされたインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる該文字列に基づいて、該文字列で特定されるインタフェースにより表されるデータ構造における特定の位置のデータを該リクエストに含まれていたデータに更新するデータ更新手段と、

を更に有することを特徴とするＷｅｂアプリケーション生成装置。

【請求項１４】　請求項１３記載のＷｅｂアプリケーション生成装置であって、
前記リクエストに前記データと共に含まれている文字列は、前記ＨＴＭＬ文の生成の基礎とされたインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる文字列に、更に該位置のデータ群が有するデータ構造を表すインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる文字列を組み合わせて成り、
前記データ更新手段は、前記文字列で特定されるデータ構造における特定の位置に更に有しているデータ構造における特定の位置のデータを前記リクエストに含まれていたデータに更新する、
ことを特徴とするＷｅｂアプリケーション生成装置。

【請求項１５】　クライアントとの間で各種のデータを授受するサーバ側で、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する方法であって、
前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得し、
前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得し、
前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する、
を有することを特徴とするＷｅｂアプリケーション生成方法。

【請求項１６】　クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを記録した記録媒体であって、
前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得する制御と、
前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する制御と、
前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生

成する制御と、
をコンピュータに行なわせる制御プログラムを記憶した記憶媒体。

【請求項１７】　クライアントとの間で各種のデータを授受するサーバ側に設けられ、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成装置であって、
処理の内容が定義されている処理ロジックが格納される処理ロジック格納手段と、
前記処理ロジックの実行条件が格納される実行条件格納手段と、
前記処理ロジックの名称となる文字列を生成する処理ロジック名生成手段と、
前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成するＨＴＭＬ文生成手段と、
を有することを特徴とするＷｅｂアプリケーション生成装置。

【請求項１８】　請求項１７記載のＷｅｂアプリケーション生成装置であって、
前記処理ロジック格納手段には複数の前記処理ロジックが格納され、
複数の前記処理ロジックのそれぞれの前記実行条件のうちのいずれかが同一であるときに、該実行条件が同一である該処理ロジックを順に実行する処理ロジックを生成する処理ロジック生成手段を更に有し、
前記文字列生成手段は、複数の前記処理ロジック、及び前記処理ロジック生成手段により生成された処理ロジックに対して各々異なる名称となる文字列を生成し、
前記ＨＴＭＬ文生成手段は、前記文字列を用いて前記処理ロジック生成手段により生成された処理ロジックを呼び出すＨＴＭＬ文であって、同一であった前記実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する、
ことを特徴とするＷｅｂアプリケーション生成装置。

【請求項１９】　クライアントとの間で各種のデータを授受するサーバ側で、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成方法であって、
処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成し、
前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する、
ことを特徴とするＷｅｂアプリケーション生成方法。

【請求項２０】　クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを記録した記録媒体であって、
処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する制御と、
前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する制御と、
をコンピュータに行なわせる制御プログラムを記憶した記憶媒体。

【請求項２１】　コンピュータに、
入力内容をデータオブジェクトに変換するステップと、
前記データオブジェクトの種類と、コマンドと、前記データオブジェクトの種類とコマンドの組合せを各処理ルーチンにマッピングする外部定義ファイルと、に基づいて処理ロジック内の処理ルーチンから適当な処理ルーチンを決定するステップと、
を実行させるためのプログラム。

【請求項２２】　クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムであって、
前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得する制御と、
前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する制御と、
前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御と、
をコンピュータに行なわせるための制御プログラム。

【請求項２３】　クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御

を該コンピュータに行なわせる制御プログラムであって、

処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する制御と、

前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する制御と、

をコンピュータに行なわせるための制御プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、企業の基幹業務などを処理するビジネスアプリケーションソフトウェアの開発効率と保守性を向上させる技術に関する。

【０００２】

【従来の技術】企業業務におけるデータエントリシステム、ワークフローシステムなどのビジネスアプリケーションでは、図４７に示すように、データベースサーバ５４０２の管理するデータを、クライアント５４０３からアプリケーションサーバ５４０１を介して操作する、という方式がとられてきた。近年、このようなビジネスアプリケーションはＷｅｂアプリケーションとして実現されるようになった。

【０００３】そして、Ｗｅｂブラウザを制御する役割を果たすアプリケーションサーバ５４０１の開発には、従来より様々な試みがなされてきている。代表的なものとしてＳｅｒｖｌｅｔ（サーブレット）、ＪＳＰ（Ｊａｖａ（登録商標）　Ｓｅｒｖｅｒ　Ｐａｇｅｓ）を用いたアプリケーションサーバ５４０１がある。

【０００４】Ｓｅｒｖｌｅｔを用いたアプリケーションサーバ５４０１は、図４８に示すように；画面表示を規定するＨＴＭＬ（Ｈｙｐｅｒ　Ｔｅｘｔ　Ｍａｒｋｕｐ　Ｌａｎｇｕａｇｅ）５５０１、画面データ５５０２、ロジック５５０３（画面の入力のハンドリング、入力データのチェック、データの加工、データベースサーバ５４０２へのデータの伝達などを行う）を一つのモジュールとして記述して実現される。

【０００５】また、ＪＳＰを用いたアプリケーションサーバ５４０１は、図４９に示すように、Ｊａｖａ　Ｂｅａｎ（Ｊａｖａ、Ｊａｖａ　Ｂｅａｎｓは、サンマイクロシステムズ・インコーポレーテッドの登録商標）というオブジェクトに格納された画面データ５６０２とＨＴＭＬ５６０１とロジック５６０３から成るモジュールとして記述するか、あるいは、ＨＴＭＬ５６０１’からなるモジュールと、画面データ５６０２’とロジック５６０３’をＪａｖａ　Ｂｅａｎオブジェクトに格納するように記述して実現される。

【０００６】一般にビジネスアプリケーションでは扱うデータが多量となるため、アプリケーションサーバ５４０１は、データ（画面データ）、ロジック、画面（ＨＴＭＬ等）を各モジュールに分けて記述して実現し、それぞれのモジュールを再利用したいという要求がある。しかしながら、図４８や図４９に示したように、Ｓｅｒｖｌｅｔ やＪＳＰ等の従来技術を用いたアプリケーションサーバ５４０１では、これらのモジュールの分離が困難であった。

【０００７】

【発明が解決しようとする課題】そこで本発明の課題は、Ｗｅｂアプリケーションを実行するアプリケーションサーバを開発する際にデータ、ロジック、画面の各モジュールに分けて記述するフレームワークを提供することにより、Ｗｅｂアプリケーションの開発効率と保守性を向上することにある。

【０００８】

【課題を解決するための手段】上述した課題を解決するために、本発明ではＷｅｂアプリケーションをカスタムタグ付きＨＴＭＬ、データオブジェクト、ロジックに分け、カスタムタグ付きＨＴＭＬとロジックの間をマッピングファイルによってマッピングし、両者間のデータのやり取りにデータオブジェクトを用いるようにした。

【０００９】本発明の一態様によれば、Ｗｅｂページの入力内容をデータオブジェクトに変換する入力内容変換手段と、複数の処理ルーチンを備える処理ロジックと、前記データオブジェクトの種類とコマンドの組合せを前記各処理ルーチンにマッピングする第１の外部定義ファイルと、前記データオブジェクトの種類とコマンドと前記第１の外部定義ファイルに基づいて前記処理ロジックの備える処理ルーチンから適当な処理ルーチンを決定する処理ルーチン決定手段と、前記処理ロジックの処理結果とデータオブジェクトの種類の組み合わせを表示用コンポーネントにマッピングする第２の外部定義ファイルと、を備える。

【００１０】また、本発明の別の一態様によれば、クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得する手段と、前記データオブジェクトが有するデータについてのＷｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する手段と、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する手段と、を備える。

【００１１】また、本発明の更なる別の一態様によれば、処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する手段と、前記文字列を用いて前記処理ロ

ジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントがクライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する手段と、を備える。

【００１２】このような構成をとることにより、Ｗｅｂアプリケーションシステムにおいて画面表示を規定するＨＴＭＬ、データオブジェクト、及び処理の内容を規定するロジックの連携がそれぞれ疎となることになり、各モジュールの再利用性が向上し、開発効率と保守性を上げることができる。

【００１３】

【発明の実施の形態】以下、本発明の実施の形態を図面を参照しながら説明する。

【００１４】図１は、本発明の概略を示す図である。本発明を実施するＵＪＩ（宇治）なるアプリケーションサーバは、図示するように、画面表示を規定するＨＴＭＬ１０１、画面データ１０２、ロジック１０３の３つのモジュールから構成される。尚、ＨＴＭＬ１０１は画面表示を規定できるものなら何れのものでも良い。

【００１５】図２は、本発明の原理構成を示す図である。図１のＨＴＭＬ１０１に対応するものが入力用ＪＳＰ２０１及び表示用ＪＳＰ２０２であり、図１の画面データ１０２に対応するものが入力用Ｊａｖａ　Ｂｅａｎ２０３及び表示用Ｊａｖａ　Ｂｅａｎ２０４、図１のロジック１０３に対応するのがユーザロジック２０５である。入力用ＪＳＰ２０１は画面をデータにアサインする機能を備え、表示用ＪＳＰ２０２はデータを画面に表示する機能を備える。ＵＪＩエンジン２０６は、入力用ＪＳＰ２０１の画面データを例えばＪａｖａ　Ｂｅａｎなどのデータオブジェクトに変換する（（１）、（５）、（６））。すなわち、入力用ＪＳＰ２０１の画面データは入力用Ｊａｖａ　Ｂｅａｎ２０３に変換される。ユーザロジック２０５は複数の処理ルーチンから構成されており、ＵＪＩエンジン２０６がデータオブジェクト（入力用Ｊａｖａ　Ｂｅａｎ２０３）の種類及びコマンドと外部のマッピングファイル２０７とに基づいて、適当な処理ルーチンを決定する（（５）、（６）、（７））。ユーザロジック２０５では、決定された処理ルーチンを用いてデータオブジェクトである入力用Ｊａｖａ　Ｂｅａｎ２０３を処理し（２）、表示用Ｊａｖａ　Ｂｅａｎ２０４として出力する（（３））。表示用Ｊａｖａ　Ｂｅａｎ２０４はＵＪＩエンジン２０６によって表示用ＪＳＰ２０２の画面データに変換され（（４）、（５）、（６））、また処理結果とデータオブジェクトである表示用Ｊａｖａ　Ｂｅａｎの種類の組み合わせを外部のマッピングファイル２０７から探し出し、表示すべき部品を決定する。このように、本発明はデータオブジェクトが画面を規定するＨＴＭＬとロジックとの間に介在するような構成をとるので、画面とロジックとを疎に連結することを可能とする。

【００１６】図３は、本発明の第１の実施例のシステム構成を示す図である。クライアントからのリクエスト３０１はフロントコンポーネント３０２で受信される。そして、リクエスト３０１に含まれるリクエストデータはＵＪＩエンジン３０４で自動解析され、データオブジェクトである入力用Ｊａｖａ　Ｂｅａｎ３０３が自動生成される。ＵＪＩエンジン３０４では、更に、コマンドマッピング３０５を参照し、リクエストデータから生成されたデータオブジェクトの種類とコマンドとからユーザロジック３０７での処理を決定する。ユーザロジック３０７では、生成されたデータオブジェクトを読み込み、該決定に従って処理を行い、処理結果として表示用Ｊａｖａ　Ｂｅａｎ３０９を設定する。ここで、ＵＪＩエンジン３０４はページマッピング３０６を参照し、表示用Ｊａｖａ　Ｂｅａｎ３０９から表示用ＪＳＰ３１０、すなわち画面表現が決定される。尚、表示用ＪＳＰ３１０にはページレイアウトを規定するテンプレート３０８がインクルードされ、このテンプレート３０８に従って表示部品が配置されるようになっている。

【００１７】図３では本発明の第１の実施例のシステム構成を示しその動作概要を述べたが、図４を用いて、本発明の第１の実施例のシステムの具体的な動作を説明する。図４は、ユーザが何処にいるかを示す「行き先掲示板」というＷｅｂアプリケーションの様子を示したものである。ユーザを一覧表示するユーザ一覧表示画面４０１、ユーザの状態を編集するユーザ状態編集画面４０２、ユーザの行き先候補を編集するプロファイル編集画面４０３、新規ユーザを登録するユーザ登録画面４０４はそれぞれ、クライアントでＷｅｂブラウザによって表示されるものである。ユーザ一覧表示画面４０１において、ログインボタン４０５が押し下げられる（クリックされる）とユーザ状態編集画面４０２に遷移する。ユーザ一覧表示画面４０１で、ユーザ登録ボタン４０６が押し下げられるとユーザ登録画面４０４に遷移する。同様に、各画面４０２～４０４の各ボタンが押し下げられると、対応する図４に示した矢印のように画面が遷移する。このとき、ログイン４０５、ユーザ登録ボタン４０６、変更ボタン４０７、ログアウトボタン４０８等の画面の各ボタンはユーザがシステムに対して処理を要求するためのコマンドに相当する。

【００１８】また、例えばユーザ「松塚」がユーザ状態として表示できる「自席」、「出張」、「年次」という３状態の他に「会議室」という状態を追加したい場合には、ユーザ状態編集画面４０２において、プロファイル編集ボタン４０９を押し下げる。すると画面表示はプロファイル編集画面４０３に遷移する。ここで、ユーザ「松塚」が「追加」の欄に例えば「会議室」と入力し、変更ボタン４１０を押し下げると新たな状態が加えられ

たプロファイル編集画面が表示される。

【００１９】以下、図５及び図６で、図４のアプリケーションのシステム構成とその動作概要を説明する。

【００２０】図５は、クライアントからのリクエストが受信されてから、処理が完了するまでの動作概要を示すものである。Ｗｅｂブラウザに表示される、ユーザ一覧表示画面５０１、ユーザ状態編集画面５０２、プロファイル編集画面５０３、ユーザ登録画面５０４は、それぞれ図４の画面に対応している。それぞれの画面から、ログイン、ユーザ登録、プロファイル編集、ログアウト、変更、ユーザ追加、キャンセル等のコマンドがＵＪ１５０６に伝達される（実線矢印）と、それぞれの画面データがデータオブジェクトであるＪａｖａ　Ｂｅａｎ５０５に自動的に変換される（点線矢印）。また、ＵＪ１５０６は、コマンドマッピング５０７（後述）を参照し、伝達されたコマンドとデータオブジェクトの種類との組み合わせに基づいて分岐すべき処理を処理５０８～５１２の中から選択する。そして選択された処理にデータオブジェクトを渡す。データオブジェクトが渡されると、選択された処理の内容に沿って、そのデータオブジェクトが処理される。

【００２１】図６は、データオブジェクトの処理が終わり、クライアントへレスポンスが返されるまでの動作概要を示すものである。処理が終わり、処理結果としてのＪａｖａ　Ｂｅａｎ６０１が作成される。そして、ＵＪ１５０６において、ページマッピング６０２（後述）が参照され、処理結果（成功、失敗、完了等）と作成されたＪａｖａ　Ｂｅａｎ６０１との組み合わせから表示すべきページがページ５０１～５０４の中から決定される。表示すべきページには作成されたＪａｖａＢｅａｎ６０１が渡され、それをもとに新たなページが表示される。

【００２２】図７（ａ）、（ｂ）にコマンドマッピングの一部、図７（ｃ）にページマッピングの一部を示す。コマンドマッピングは入力データオブジェクトの種類、コマンド、処理から成るテーブルで構成される。また、ページマッピングは出力データオブジェクトの種類、処理結果、画面から成るテーブルで構成される。

【００２３】図７（ａ）、（ｂ）に示すように、コマンドマッピングによって、入力データオブジェクトの種類とコマンドとの組み合わせから分岐すべき処理が決定される。図７（ａ）は入力データオブジェクトの種類が同一で、コマンドの種類が異なる場合のコマンドマッピングを示したものである。図７（ａ）より、入力データオブジェクトの種類が「ユーザ状態」で、コマンドが「プロファイル編集」の場合、プロファイル変更処理に処理を分岐させることが決定されることがわかる。また、図７（ｂ）は、入力データオブジェクトの種類が異なり、コマンドが同一な場合のコマンドマッピングを示したものである。図４に示したように「変更」コマンドは様々な画面で用いられる。しかし、コマンドが同一であっても、コマンドが発生したときに分岐すべき処理はそれぞれ異なる。図４のユーザ状態編集画面４０２において「変更」コマンドが発生した場合は、ユーザ状態変更処理に処理を分岐させなければならず、プロファイル編集画面４０３において「変更」コマンドが発生した場合はプロファイル変更処理に処理を分岐させなければならない。図７（ｂ）に示すコマンドマッピングでは、入力データオブジェクトの種類が「ユーザ状態」で「変更」コマンドが発生するとユーザ状態変更処理に処理を分岐させ、入力データオブジェクトの種類が「プロファイル編集」で「変更」コマンドが発生したときにはプロファイル編集処理に分岐させるように規定することができる。このように、本発明のコマンドマッピングは入力データオブジェクトの種類とコマンドの組み合わせから分岐すべき処理を決定するため、コマンドが同一であっても、入力データオブジェクトの種類から分岐すべき処理を適切に選択することができる。

【００２４】また、図７（ｃ）に示すページマッピングによって、出力データオブジェクトの種類と処理結果との組み合わせから、表示すべき画面が決定される。ページマッピングにおいてもコマンドマッピングと同様に、出力データオブジェクトの種類と処理結果との組み合わせで画面が決定されるため、処理結果が同一であっても、出力データオブジェクトの種類から表示すべき画面を適切に選択することが可能である。

【００２５】尚、コマンドマッピング、ページマッピングは、外部定義ファイルで、開発者が自由に設定できるものである。これにより開発における柔軟性を高めることができる。

【００２６】以上、本発明の第１の実施例のシステム全体の動作を説明したが、次に、各動作間についてより具体的に説明する。

【００２７】図８は、クライアントからのリクエストデータを入力用Ｊａｖａ　Ｂｅａｎに変換させる（図３の３０１、３０３間）ためのプログラム記述を示す図である。この記述により、データ入力用ＨＴＭＬページ８０１にデータが入力されると、データ入力用ＨＴＭＬページ８０１の特定項目（LoginBean ）に対応したクラス名（図８の①）であって、各入力欄の名前（name）をプロパティ（図８の②、②’）として持つデータオブジェクト入力用Ｊａｖａ　Ｂｅａｎ８０２が生成される。

【００２８】図９は、実行すべきロジックの処理をコマンドマッピングを用いて決定する（図３の３０５、３０７間）部分のプログラム記述を示す図である。コマンドマッピング９０２には、データ入力用ＨＴＭＬページ９０１の特定項目に対応したクラス名（LoginBean ）と、押し下げられたボタン（submit）の名前（name）の組み合わせが、ユーザロジック９０３（ハンドラ）のメソッドにマッピングされている。これにより、例えばデータ

入力用ＨＴＭＬページ９０１でログインボタンが押し下げられると、ユーザロジック９０３のloginメソッドが実行されることになる（図９の①）。同様に、データ入力用ＨＴＭＬページ９０１でパスワード変更ボタンが押し下げられると、ユーザロジック９０３のchangePasswordメソッドが実行されることになる（図９の②）。

【００２９】図１０は、ユーザロジックが表示用データオブジェクトを設定する（図３の３０７、３０９間）部分のプログラム記述を示す図である。ユーザロジック（ハンドラ）１００１は、データベース１００２などを使用して処理を行い、処理の結果として表示が必要である値と処理結果とを表示用データオブジェクト（Ｊａｖａ Ｂｅａｎ）１００３に設定する。

【００３０】図１１は、ページマッピングを介して表示すべきページを決定する（図３の３０９、３１０間）部分のプログラム記述を示す図である。ページマッピング１１０２では、クラス名（UserBean）とユーザロジック（ハンドラ）で設定される処理結果の組み合わせが表示すべきビュー（この場合ＪＳＰ）にマッピングされている。これにより、表示用Ｊａｖａ Ｂｅａｎ１１０１に設定された処理結果が例えばsucceededの場合、login-succeeded.jsp １１０３が選択され（図１１の①）、これと図３のテンプレート３０８とが合わされて表示画面が決定される。同様に、表示用Ｊａｖａ Ｂｅａｎ１１０１に設定された処理結果がfailedの場合、login-failed.jsp１１０４が選択され（図１１の②）、これと図３のテンプレート３０８とが合わされて表示画面が決定される。

【００３１】図１２は、ＪＳＰから表示画面を出力する（図３の３１０）部分のプログラム記述を示す図である。表示用ＪＳＰ１２０１は、データ取得用のタグを使用して表示用Ｊａｖａ Ｂｅａｎ１２０２からデータを取得し、取得されたデータを出力ＨＴＭＬ１２０３として出力する。

【００３２】図１３は、表示画面に複数の表示部品を配置する場合のテンプレート（図３の３０８、３１０間）について説明する図である。ユーザロジック（ハンドラ）１３０１は、同時に複数の表示用Ｊａｖａ Ｂｅａｎを設定するようにすることができる。例えば、図１３のようにバナー用のＪａｖａ Ｂｅａｎ１３０２、メニュー用のＪａｖａ Ｂｅａｎ１３０３、コンテンツ用のＪａｖａ Ｂｅａｎ１３０４がユーザロジック１３０１で設定されると、それぞれのＪａｖａ Ｂｅａｎに対応するＪＳＰがそれぞれのＪａｖａ Ｂｅａｎからデータを取得してそれぞれの表示画面を作成する。テンプレート１３０５にはＪＳＰごとに表示位置が規定されているため、作成された各表示画面がテンプレート１３０５に基づいて配置され、出力ＨＴＭＬ１３０６が出力される。

【００３３】上述のように、本発明第１の実施例のシステムではＨＴＴＰ（Ｈｙｐｅｒ Ｔｅｘｔ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）リクエストをデータオブジェクトに変換して処理し、処理結果のデータオブジェクトをレスポンスに変換してクライアントに返す仕組みについて詳細に述べた。次に、同一の仕組みを利用して、ＸＭＬ（ｅＸｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）ファイルを処理する本発明の第２の実施例のシステムについて述べる。図１４、１５にそのシステム構成を示す。

【００３４】図１４は、アプリケーションサーバがＸＭＬファイルを受信したときにそのＸＭＬファイルがロジック（ハンドラ）で処理されるまでを示す図であり、図１５はロジック（ハンドラ）でそのＸＭＬファイルの処理が終わり、クライアントへレスポンスが返されるまでを示す図である。

【００３５】まず、図１４において、サーバがＸＭＬインスタンス１４０１を受信すると、そのＸＭＬをＸＭＬ Ｄａｔａ Ｂｉｎｄｉｎｇエンジン１４０４で処理を行ってデータオブジェクト（Ｂｅａｎインスタンス１４０５）を作成する。この際、受け取ったＸＭＬインスタンス１４０１からオブジェクトの種類を特定することができるため、種類の異なるＸＭＬを同一エンジンで受け取ることができる。ここでは、オブジェクトの種類は、ＸＭＬマッピングファイル（不図示、後述）を利用して特定される。また、ＵＪＩエンジン１４０３は、コマンドマッピング１４０２を参照することによって、ＸＭＬのタグからロジック（ハンドラ）で実行すべき処理（メソッド）を決定する。

【００３６】図１５において、ロジック（ハンドラ）１５０１では、ＨＴＴＰリクエストのときと全く同様にして送信用のデータオブジェクト（Ｂｅａｎインスタンス１５０２）が作成される。これが再び、ＸＭＬ Ｄａｔａ Ｂｉｎｄｉｎｇエンジン１４０４でＸＭＬに変換されて送信用ＸＭＬ１５０３が出力される。

【００３７】尚、データオブジェクトを生成するＸＭＬ Ｄａｔａ Ｂｉｎｄｉｎｇエンジン１４０４は、ファクトリクラスから呼び出される。そのため、ファクトリクラスをＢｉｎｄｉｎｇエンジンの種類毎に用意することで、柔軟に複数のＢｉｎｄｉｎｇエンジンに対応することができる。

【００３８】図１６及び図１７は図１４及び図１５に示したシステムの動作概要を示すものである。図１６は、図１４に対応し、アプリケーションサーバがＸＭＬファイルを受信してからロジック（ハンドラ）で処理されるまでを示した図であり、図１７は図１５に対応し、ロジックで処理が終了してから、クライアントへレスポンスを返すまでを示した図である。

【００３９】図１６において、注文伝票１６０１、出荷伝票１６０２等のＸＭＬファイルがアプリケーションサーバに対して送信される（実線矢印）と、それぞれの画

面データがデータオブジェクトであるＪａｖａ　Ｂｅａｎ１６０５に自動的に変換される（点線矢印）。また、ＵＪＩ１６０３において、ＸＭＬマップファイル１６０６を利用してＸＭＬインスタンスのオブジェクトの種類が特定され、ＸＭＬＤａｔａ　Ｂｉｎｄｉｎｇエンジン１６０４で受信したＸＭＬに対応するデータオブジェクトが作成される。またＵＪＩ１６０３で、ＸＭＬのタグとコマンドマッピング１６０７から、分岐すべき処理を処理１６０８、１６０９、１６１０等の中から選択する。そして、選択された処理にデータオブジェクトを渡す。データオブジェクトが渡されると、選択された処理の内容に沿ってそのデータオブジェクトが処理される。

【００４０】処理が終わると、図１７において、処理結果であるＪａｖａ　Ｂｅａｎ１７０１がＵＪＩ１６０３に渡される。このＪａｖａ　Ｂｅａｎ１７０１は、ＸＭＬＤａｔａ　Ｂｉｎｄｉｎｇエンジン１６０４で送信用ＸＭＬインスタンス１７０２に変換される。

【００４１】このように第２の実施例のシステムでは受信したＸＭＬをＸＭＬ　ＤａｔａＢｉｎｄｉｎｇエンジンでデータオブジェクトに変換してからロジック（ハンドラ）に渡すので、ＨＴＴＰリクエストの時と同じ処理を実現する場合、第１の実施例のシステムのロジック（ハンドラ）と同一のロジック（ハンドラ）のメソッドをそのまま利用することができる。

【００４２】以上、本発明の第１及び第２の実施例のシステム構成、システム動作について詳細に説明したが、次にこれらの実施例に共通するシステムの動作の仕組みについて説明する。

【００４３】本発明のシステムを構成するオブジェクトは、各々が処理されるときの時間軸上における相対的な位置関係に基づいてシステム、アプリケーション、セッション、リクエストの４段階の時間的スコープに分類することができる。図１８（ａ）に示すように、システム１８０１は、システム全体で一つ存在するもので、アプリケーションサーバの起動からサーバの終了までをそのスコープとする。また、アプリケーション１８０２は、システム内の各アプリケーション毎に１つ存在し、アプリケーションの開始から終了までをそのスコープとする。セッション１８０３は、クライアント毎に一つずつ存在するもので、クライアントの接続から切断（またはタイムアウトなど）までをそのスコープとする。そして、リクエスト１８０４は、あるクライアントからの一回のリクエストにつき一つ存在するもので、クライアントからのリクエストを処理してレスポンスを返すまでをスコープとする。図１８（ｂ）に、各オブジェクト間の相関関係を示す。システム１８０１内に複数のアプリケーション１８０２が存在し、各アプリケーション１８０２に接続する複数のクライアントからのセッション１８０３が存在する。そして、クライアントからのリクエストは、予め用意されるハンドラ１８０５で処理される。

尚、図１８　（ａ）に示すリクエスト１８０４は、図１８（ｂ）の各オブジェクトが実行される際の１スレッドに対応する。

【００４４】このようにシステムを構成するオブジェクトが段階的な時間的スコープを持つことによって、クライアントからのリクエストを処理する場合に、より大きなスコープを持つオブジェクトから小さなオブジェクトに分岐（dispatch）して処理を進めていくことができる。すなわち図１９に示すように、フロントコンポーネント１９０１にクライアントからリクエストが送信されてくると、システムに一つ存在するディスパッチャ１９０２が対応するアプリケーション１９０３に処理を分岐する。更にアプリケーション１９０３は、リクエストを送信してきたクライアントに対応するセッション１９０４に処理を分岐し、セッション１９０４は、リクエストに対応するハンドラ１９０５に処理を分岐する。

【００４５】本発明のシステムでは、クライアントからのリクエストをより大きなスコープから小さなスコープに分岐して処理していくため、アプリケーション、セッション、ハンドラの各オブジェクトをユーザ拡張可能またはユーザ定義可能としておくと次のような効果が得られる。（尚、本実施例のシステムでは、アプリケーション１９０３とセッション１９０４をユーザ拡張可能、フロントコンポーネント１９０１、表示用ページ１９０６、ハンドラ１９０５、コマンドマップ１９０９、ページマップ１９１０、ＲｅｓｐｏｎｓｅＢｅａｎ１９０８及びＲｅｑｕｅｓｔＢｅａｎ１９１１をユーザ定義可能として実装した。）

ユーザ拡張可能またはユーザ定義可能なオブジェクトに対して、SingleThreadModel １９１２というインタフェースを実装させると（図１９（ｂ））、そのオブジェクトの動作が同時に最大１スレッドまで制限される。この機能を利用すれば、それぞれのスコープの動作がスレッドセーフとなり、すなわち、例えば同一データ編集などのデータ処理は、同時に複数行わせないようにすることができる。このようにすることを「シングルスレッド動作制限」という。前述の「行き先掲示板」アプリケーション（図４）において、ユーザ変更とプロファイル編集は同一データを対象とするので、スレッドセーフにしておく必要がある。この場合、図２０（ａ）に示すように実装すればよい。すなわち、メソッド：ユーザ変更２００７、メソッド：プロファイル編集２００８を含むハンドラ２００４にSingleThreadModelインタフェースを実装させる。ここで、メソッド：ログイン処理２００５、メソッド：ログアウト処理２００６、メソッド：ユーザ登録、は特にスレッドセーフにしなくてよいため、ハンドラ２００４’、２００４’’にはSingleThreadModelインタフェースは実装させない。同様に、セッション、アプリケーションにおいても、スレッドセーフにしたい場合、SingleThreadModel インタフェースを実装させれ

ばよい（２００２’、２００３’）。SingleThreadModelインタフェースを実装する場合の記述の仕方は、図２０（ｂ）のようにすればよく、同図には、セッション２００３’にSingleThreadModel インタフェースが実装された場合が示されている。

【００４６】また、ユーザ拡張可能またはユーザ定義可能なオブジェクトである、アプリケーション、セッション、ハンドラオブジェクトに前処理（preprocessor）、後処理（postprocessor ）インタフェースを実装させることができる。これにより、各オブジェクトの本番処理の実行の前後に処理を付け加えることができる。すなわち、図２１のシーケンス図に示すように、クライアントのリクエストの本番処理（ハンドラメソッド）（図２１の②）の前処理（図２１の①）、後処理（図２１の③）を加えることができ、同様に、セッション（図２１の⑤）に対する前処理（図２１の④）、後処理（図２１の⑥）、アプリケーション（図２１の⑧）に対する前処理（図２１の⑦）、後処理（図２１の⑨）を追加することができる。

【００４７】この機能を利用して、例えば前処理（図２１の①）では、ロジックのハンドラメソッドの本番処理（図２１の②）の前に予め状態管理による判断を行ってハンドラメソッドの処理をスキップするチェックルーチンを設けるようにすることが可能である。また、後処理（図２１の③）では、前処理（図２１の①）または本番処理（図２１の②）でエラーが出たときにそのエラーを回復させるためのエラーハンドリングを設けておくことができる。これらは、エラーが出なかったときでも、共通の後処理を記述するために利用することもできる。同様に、各セッションについて、１クライアントに対する共通の前処理、後処理を記述することができ、アプリケーションでは複数クライアントに対する共通の前処理、後処理を記述することができる。各前処理、後処理はそれぞれのスコープのオブジェクトに別々に実装することができるため、きめ細かい制御ができる。

【００４８】以上のように、本発明のシステムの動作の仕組みを用いれば、スレッドセーフを実現させたり、各スコープにおける処理に前処理、後処理を追加することで処理の前判断やエラー処理が柔軟に行なえるようになるという効果を得ることができる。

【００４９】次に、本発明のアプリケーションサーバにおけるロジックのハンドラからブラウザにリクエストを行うための仕組みについて説明する。

【００５０】Ｗｅｂアプリケーションでは、例えば、処理Ａのリクエストがクライアントから送られてきたが、何らかのことが起こったために、処理Ａを続行するか否かをユーザ（クライアント）に問い合わせ、その回答に基づいて処理Ａを続行するか若しくは異なる処理を行うようにするという場合がある。このように、処理の途中でユーザ（クライアント）に問い合わせが行なわれ、この問い合わせの結果に基づいて処理を行っていく場合、一般のＷｅｂアプリケーションシステムでは、図２２（ａ）に示すようなシーケンスで行っている。処理Ａのリクエスト（図２２（ａ）の①）がサーバに送信されると、サーバではそれを受信し、ハンドラ１が処理を始める。すると、処理Ａは全て終了していないが、ユーザに処理Ａを続行するかどうか質問する（図２２（ａ）の②）。クライアントには、２７０１のような画面が表示され、ユーザが「はい」「いいえ」の何れかの回答を返す（図２２（ａ）の③）。それに基づき、ハンドラ２が処理を始める。

【００５１】このように、一般のＷｅｂアプリケーションでは、処理Ａが全て完了していなくてもハンドラの処理を中断しなくてはならない場合がある。また、ユーザの回答が「はい」、「いいえ」などの場合、何の質問に対する「はい」、「いいえ」なのかを規定した中間処理のためのマッピング定義が必要である。

【００５２】そこで、本発明のシステムに、以下に述べるようなアプリケーションサーバにおけるロジックのハンドラからブラウザにリクエストを行う仕組みを導入し、一般のＷｅｂアプリケーションシステムを改善した。

【００５３】図２３（ａ）に示すように、本発明のシステムでは、サーバにおいて、ハンドラ呼び出し前にスレッド緩衝オブジェクト（シンクロナイザ）を設け、ハンドラの動作を監視するようにする。これにより、処理Ａのリクエスト（図２３（ａ）の①）がサーバに送信されると、サーバではシンクロナイザが受信し、ハンドラ１に渡され（図２３（ａ）の②）、ハンドラ１が起動して処理が始まる。そこで、処理Ａが全て終了していないが、ユーザに処理Ａを続行するかどうか質問をする場合、ハンドラ１はクライアントにコールバックするメソッドを呼び出す（図２３（ａ）の③）。シンクロナイザでは、ハンドラのメソッドを、ブラウザ（クライアント）への返り値として返す（図２３（ａ）の④）。クライアントには、２２０１のような画面が表示され、ユーザが「はい」「いいえ」の何れかの回答を返す（図２３（ａ）の⑤）。それがシンクロナイザに受信され、ハンドラ１に渡される（図２３（ａ）の⑥）。この後、ハンドラ１は、処理を継続する。このように、スレッド緩衝オブジェクトが設けられることで、ハンドラの処理を中断させる必要がなくなる。また、サーバを実現するためのプログラムの記述においても、一般的なＷｅｂアプリケーションでは図２２（ｂ）に示すように、ハンドラ１、ハンドラ１の返り値に対するイベントを受けるハンドラ２などを記載しなくてはならないのに対し、本発明では図２３（ｂ）（ｃ）に示すように、一つのハンドラ内に条件分岐文で記述することができるため、プログラム記述的に非常に容易である。

【００５４】なお、以上までに説明した本発明の実施例

では、クライアントからの入力としてＨＴＴＰ、ＸＭＬ等を対象としたが、クライアントからリクエストであればどのようなものでもよく、また、これらの本発明の実施例では、クライアントのリクエストをＪａｖａ Ｂｅａｎというデータオブジェクトに変換したが、他のデータオブジェクトでもよい。

【００５５】以上までに説明した実施例に係る発明は、クライアントからの入力をデータオブジェクトに変換し、それを処理し、処理結果として得られるデータオブジェクトをクライアントへのレスポンスに変換して送信することを特徴とするものであり、また、サーバにおける処理を時間的スコープに分類し、スコープの大きなものから小さなものに分岐して処理を進めることを特徴とするものである。

【００５６】次に、本発明の更なる実施例について説明する。

【００５７】これより説明する実施例は、Ｗｅｂアプリケーションを実行するアプリケーションサーバを開発する際に、画面の見かけの定義（ビュー）とその画面中に埋め込まれるデータとを分離して実装するようにして、Ｗｅｂアプリケーションの開発効率と保守性とを共に向上させるものである。

【００５８】この実施例では、Ｗｅｂアプリケーションにおける表示部を、表示データオブジェクトと、画面の見かけ（ビュー）の定義とに分けて用意する。このとき、表示データオブジェクトは、表示データ自体を有し、テーブル（表）構造や木構造などのデータ構造（本実施例ではこのデータ構造を「モデル」と称することとする）を表すインタフェースであってその表示データについてのものを実装した構造体クラスとして作成する。また、ビューの定義は、この実施例においてはＪＳＰ（Java Server Pages ）を用いて記述する（ビューの定義は、単にＨＴＭＬで記述するようにしてもよい）。

【００５９】この表示データオブジェクトとビューの定義とがサーバにおいて実行されると、上述したインタフェースに対応する実行エンジンによって表示データを画面の所定位置に割り当てる（アサインする）ＨＴＭＬ文が生成される。その後、このＨＴＭＬ文がクライアントに送付されると、クライアントの備えるＷｅｂブラウザによって対応する画面表示が行なわれる。

【００６０】なお、これより説明する上述した実施例を、既に説明した他の実施例と区別するために、「第３の実施例」と称することとする。

【００６１】まず図２４について説明する。同図は、本発明の第３の実施例のシステム構成を示している。

【００６２】モデルオブジェクト３００１は、サーバ側に設けられているデータベースなどのバックエンドから取得されるクライアントに表示すべきデータを保持し、また、クライアントから送付されたデータをバックエンドに渡すために保持するオブジェクトであり、後述するモデルインタフェース３００２を実装する。なお、モデルオブジェクト３００１は、表示用モデルオブジェクト３００１ａと入力用モデルオブジェクト３００１ｂとして別々に設けることも可能であり、また、表示用と入力用とを共用する単一のモデルオブジェクト３００１として設けることも可能である。

【００６３】モデルインタフェース３００２は、特定のモデルの構造を表すインタフェースであり、例えば、テーブル構造についてはテーブルモデル用、木構造についてはツリーモデル用などというように、対象とするデータ構造に応じたインタフェースが用意される。

【００６４】フレームワークエンジン３００３は、表示用モデルオブジェクト３００１ａに実装されているモデルインタフェース３００２に基づいて、表示用モデルオブジェクト３００１ａに保持されているデータを取り出し、表示用ＪＳＰ３００４の定義に従ってＨＴＭＬ文を生成する。また、ブラウザ３００５に対して入力されたデータを解析して入力用モデルオブジェクト３００１ｂに渡す。

【００６５】表示用ＪＳＰ３００４は表示データをＨＴＭＬで表現するときの画面の見かけ（ビュー）が定義されているものであり、独自のタグを導入したＪＳＰで記述される。

【００６６】ブラウザ３００５はクライアントに装備され、表示用ＪＳＰ３００４の定義に基づいてフレームワークエンジン３００３で生成されたＨＴＭＬに従った画面表示をで行なうものである。

【００６７】フロントコンポーネント３００６は、ブラウザ３００５から発せられるリクエストを受け付けてフレームワークエンジン３００３に渡すものであり、例えばＪＳＰコンポーネントあるいはサーブレットなどを利用するが、ＨＴＴＰリクエストを受信し、その受信内容に応じたフレームワークエンジン３００３を呼び出すことができるものであればどのようなものでもよい。

【００６８】次に、図２４のモデルインタフェース３００２について、テーブル構造のモデルを表すテーブルモデルインタフェースを例に挙げて更に詳細に説明する。

【００６９】図２５は、テーブルモデルインタフェースの宣言例を示す図である。この例では、大きく分けてデータ取得のためのメソッド、データクラスの取得のためのメソッド、及びデータの（バックエンドへの）代入のためのメソッドでテーブルモデルインタフェースが定義されている。

【００７０】これらのメソッドについて更に説明すると、データ取得のためのメソッドとしてはテーブルの列数を取得するメソッド（getColumnCount）、テーブルの行数を取得するメソッド（getRowCount ）、及びテーブルの各セルのデータを取得するメソッド（getValueAt）が図２５の定義に用いられている。

【００７１】データクラスの取得のためのメソッドとし

ては、列及び行の各々について定義される「クラス」を特定する文字列を各々取得するメソッド（getColumnClass及びgetRowClass ）が図２５の定義に用いられている。この「クラス」とは、各列若しくは各行に属するデータについての表示方法（例えば表示位置やフォントの大きさといった書式等）の様式をそれぞれ定義するために使用されるものであり、これは後述するレンダラタグの” cls” 属性を示すものである。

【００７２】データの代入のためのメソッド（setValueAt）は、このテーブルモデルに対応するモデルアップデートの処理（後述）により呼び出されることによって、このテーブルモデルで示されるデータ構造に依存したデータ形式でクライアントからのリクエストの内容（セルの値及びそのセルのテーブル上の位置）がバックエンドに渡される。

【００７３】次に、本発明の第３の実施例においてＨＴＭＬ文が生成される様子を、図２６を用いて説明する。同図は、図２４に示すテーブルモデル用のフレームワークエンジン３００３に図２５に示す宣言がなされているテーブルモデルインターフェースを実現する表示用のモデルオブジェクト３００１ａ（Ａ）を与えると、フレームワークエンジン３００３が、表示用ＪＳＰ３００４に相当するＪＳＰソース（Ｂ）に基づいて生成ＨＴＭＬ文（Ｃ）を生成することを示している。なお、説明の便宜のため、図２６（Ｂ）及び図２６（Ｃ）の各行頭には行番号を付している。

【００７４】図２６の（Ｂ）ＪＳＰソースについて説明する。

【００７５】図２６（Ｂ）において、第１行はこのＪＳＰソースとモデルオブジェクト（Ａ）との対応関係を指定するものである。

【００７６】第１行を更に説明すると、“id=...”には対応するモデルオブジェクトがこのＪＳＰソース内において参照されるときの名称が指定され、“cls=...”はこのＪＳＰソースと対応するモデルオブジェクトのクラス名が指定される。

【００７７】また、第１行において、“request=... ”にはモデルオブジェクト（Ａ）のフレームワークエンジン３００３内での生存期間が指定される。これは、生成ＨＴＭＬ文（Ｃ）をクライアントに送付してブラウザ３００５に画面表示を行なわせた後に、このモデルオブジェクトに応じたリクエストがクライアントから返信される場合を考慮したものである。すなわち、trueが指定されているときには、このようなリクエストの内容を直ちにモデルオブジェクトに代入できるようにするために、ＨＴＭＬ文の生成後もモデルオブジェクト（Ａ）がフレームワークエンジン３００３内の記憶領域に保持される。一方、falseが指定されているときには、ＨＴＭＬ文の生成後はモデルオブジェクト（Ａ）のフレームワークエンジン３００３内の記憶領域で保持されない。

【００７８】図２６（Ｂ）の第２行目以降は、テーブルモデルを表示するための画面の見かけの定義がタグを用いてなされている部分であり、この部分は「ビュー」と総称されている。

【００７９】ビューの記述はビュータグを用いて行なわれる。第２行目にはテーブルモデルのためのビュータグの開始タグが示されており（<uomf:table >）、この第２行目と第１８行目のテーブルモデル用ビュータグの終了タグ（</uomf:table>）との間に幾つかのテーブル用のレンダラが記述される。

【００８０】レンダラとは、指定の表示場所でのデータの表示にどのような表示方法を用いるかを定義するものである。図２６（Ｂ）の第３行目から第１７行目にかけての記述では、テーブルモデルのためのレンダラの定義が（ａ）から（ｅ）の計５つなされている。これらのテーブルレンダラの定義を簡単に説明すると、（ａ）はテーブルタグ（<table> 及び</table>）を生成させるためのもの、（ｂ）はテーブルの行の表示方式を定義するもの、（ｃ）はテーブルにおける通常の列の表示方式を定義するもの、（ｄ）はテーブルにおける各列の見出しが示されるセル（ヘッダセル）の表示方式を定義するもの、（ｅ）はテーブルにおけるデータ入力（データ更新）の可能なセルについての表示方式を定義するものである。

【００８１】各レンダラには、テーブルモデル用レンダラタグの開始タグ（<uomf:tableRenderer>）がその最初の行に記述され、これに対応する終了タグ（</uomf:tableRenderer>）がその最後の行に記述される。そして、このレンダラタグの開始タグと終了タグとに挟まれた行の記述によってデータの表示方法が定義される。このデータの表示方法についての定義を構成する個々の要素をレンダラエレメントと称している。

【００８２】レンダラタグでは“type”と”cls ”とについての指定がなされる。これらは、レンダラに定義された表示方法で表示を行なう画面上の場所を特定するためのものである。これらの指定内容は扱うモデルの種類によって予め指定される。

【００８３】“type”には、本実施の形態におけるテーブルモデルでは、table （表全体）、row （行）、column（列）のいずれかが指定され、これらによりレンダラに定義された表示方法で表示を行なうテーブルの場所が特定される。

【００８４】“cls ”は、本実施の形態におけるテーブルモデルでは、上述した“type”がrow 又はcolumnに指定されているときに指定が可能である（指定のないこともある）。これは、前述したテーブルモデルインタフェースの宣言例（図２５）におけるgetRowClass及びgetColumnClassの両メソッドと連動しているものであり、例えば列の表示を行なう際にはテーブルモデルインタフェースのgetColumnClassメソッドが呼び出される。そし

て、ビューのレンダラの定義が参照され、“type”がcolumnであって、“cls”が呼び出されたメソッドについての戻り値に対応しているレンダラでの表示方法の定義が、その列の表示のための表示方法としてＨＴＭＬ文の生成に使用される。なお、呼び出されたメソッドについての戻り値が無い（ヌルである）ときには“cls”の指定のないレンダラについての表示方法の定義が使用される。

【００８５】“cls ”は、本実施の形態におけるテーブルモデルでは、具体的にはheader若しくはeditableが指定される。

【００８６】レンダラにおける表示方法の定義は周知のＨＴＭＬのタグを用いて記述されるが、ここにレンダラエレメントタグという特殊なタグを導入する。図２６（Ｂ）においては、<uomf:children> 及び<uomf:value>がレンダラエレメントタグである。<uomf:children>は、このタグの位置に他のレンダラでの表示方法の定義を展開する動作をフレームワークエンジンに行なわせ、<uomf:value>は、このタグの位置にモデルインタフェースから取得した値（表示データ）を表示させるようにする動作をフレームワークエンジンに行なわせる。

【００８７】これらのタグのより詳細な動作は扱うモデルの種類によって予め規定される。本実施の形態におけるテーブルモデルでは、<uomf:children> は、前述したレンダラタグにおける“type”の指定がtable （表全体）であるときにこのタグが出現したときには行についてのレンダラ（“type”の指定がrowであるレンダラ）で定義されている表示方式をこの位置に展開して挿入し、“type”の指定がrow（行）であるレンダラでこのタグが出現したときには列についてのレンダラ（“type”の指定がcolumnであるレンダラ）で定義されている表示方式をこの位置に展開して挿入することが規定されている。また、<uomf:value>は、“type”の指定がcolumnであるレンダラにおいてのみ使用可能であり、テーブルモデルインタフェースの宣言例（図２５）で使用されているgetValueAtメソッドによって取得される値をこの位置に挿入することが規定されている。

【００８８】以下、図２６を参照しながら、フレームワークエンジンによって（Ｃ）の生成ＨＴＭＬ文が生成される様子を説明する。

【００８９】まず、フレームワークエンジンは（Ａ）のモデルオブジェクトがテーブルモデルインタフェースを実装していることを認識し、テーブルモデル用のフレームワークエンジンを起動する。そして、このテーブルモデルインタフェースのgetColumnCount及びgetRowCountの両メソッドが呼び出され、表示するテーブルの列数及び行数が認識される。この結果、ここでは３行３列のテーブルを表示されることが認識される。

【００９０】次に、（Ｂ）のＪＳＰソースにおいて、“type”がtable に指定されている（ａ）のテーブルレンダラの定義がまず参照され、（Ｃ）の生成ＨＴＭＬ文の第１行目が生成される。

【００９１】ここで、（ａ）のテーブルレンダラの定義（（Ｂ）の第４行目）にはレンダラエレメントタグ<uomf:children>が記述されており、（ａ）のレンダラタグにおける“type”はtable に指定されているので、行についてのテーブルレンダラ、すなわち（Ｂ）のＪＳＰソースにおける（ｂ）のテーブルレンダラの定義の展開が行なわれ、（Ｃ）の生成ＨＴＭＬ文の第２行目が生成される。

【００９２】ここで、（ｂ）のテーブルレンダラの定義（（Ｂ）の第７行目）にもレンダラエレメントタグ<uomf:children>が記述されており、（ｂ）のレンダラタグにおける“type”はrow に指定されているので、列についてのテーブルレンダラの定義の展開が行なわれる。但し、（Ｂ）のＪＳＰソースには、（ｃ）、（ｄ）、（ｅ）という列についてのテーブルレンダラ（“type”の指定がcolumnであるレンダラ）が３つ記述されているので、これらのうちのどのテーブルレンダラの定義の展開を行なうかが問題となる。

【００９３】このとき、フレームワークエンジンでは、（Ａ）のモデルオブジェクトから、表示させるテーブルの１行目の各列のセルの値の取得、及びテーブルモデルインタフェースのgetColumnClassメソッドの呼び出しも行なわれている。これらの処理により、ここで取得されたセルの値は、“品名”、“単価”、“個数”という３つの文字列データであり、呼び出されたメソッドについての戻り値はこれら３つのいずれの値についてもheaderであった。

【００９４】そこで、フレームワークエンジンは、このメソッドについての戻り値に基づき、（Ｂ）のＪＳＰソースにおける（ｄ）のテーブルレンダラの定義が選択され、その定義の展開を実行する。その結果、表示させるテーブルの１行目の表示方式を示すＨＴＭＬ文（（Ｃ）の生成ＨＴＭＬ文の第３、第４、第５行目）が、（ｄ）のテーブルレンダラの定義内容（（Ｂ）の第１３行目）に基づいて生成される。ここで、（ｄ）のテーブルレンダラの定義にはレンダラエレメントタグ<uomf:value>が含まれているので、この部分にはモデルオブジェクトから取得された各セルの値が挿入される。

【００９５】ここまでで、先に行なった（ｂ）のテーブルレンダラの定義（（Ｂ）の第７行目）におけるレンダラエレメントタグ<uomf:children>についての展開が完了し、そのタグに続く次の定義の記述に基づいて（Ｃ）の生成ＨＴＭＬ文の第６行目が生成され、テーブルの１行目の表示のためのＨＴＭＬ文の生成が完了する。

【００９６】次に、テーブルの２行目の表示のためのＨＴＭＬ文の生成が開始され、行についてのテーブルレンダラ、すなわち（Ｂ）のＪＳＰソースにおける（ｂ）のテーブルレンダラの定義の再度の展開が行なわれ、

（Ｃ）の生成ＨＴＭＬ文の第７行目が生成される。

【００９７】ここで、前回と同様に（ｂ）のテーブルレンダラの定義にレンダラエレメントタグ<uomf:children>の列についてのテーブルレンダラの定義への展開が行なわれる。

【００９８】このとき、フレームワークエンジンでは、（Ａ）のモデルオブジェクトから、表示させるテーブルの２行目の各列のセルの値の取得、及びテーブルモデルインタフェースのgetColumnClassメソッドの呼び出しも行なわれ、これらの処理により、ここで取得されたセルの値は、“ＸＶ－５０８０”、“１９８，０００”、“”（ヌル）という３つのデータであり、呼び出されたメソッドについての戻り値は、最初の２つのセルについての値はヌルデータであり、最後のセルについての値はeditableであった。

【００９９】そこで、フレームワークエンジンは、最初の２つのセルについてのこのメソッドの戻り値に基づき、（Ｂ）のＪＳＰソースにおける（ｃ）のテーブルレンダラの定義の展開を実行する。その結果、表示させるテーブルの２行１列目及び２行２列目の表示方式を示すＨＴＭＬ文（（Ｃ）の生成ＨＴＭＬ文の第８及び第９行目）が、（ｃ）のテーブルレンダラの定義内容（（Ｂ）の第１０行目）に基づいて生成される。ここで、（ｃ）のテーブルレンダラの定義にもレンダラエレメントタグ<uomf:value>が含まれているので、この部分にはモデルオブジェクトから取得された各セルの値が挿入される。

【０１００】更に、フレームワークエンジンは、上述した最後のセルについてのこのメソッドの戻り値に基づき、（Ｂ）のＪＳＰソースにおける（ｅ）のテーブルレンダラの定義の展開を実行する。その結果、表示させるテーブルの２行３列目の表示方式を示すＨＴＭＬ文（（Ｃ）の生成ＨＴＭＬ文の第１０行目）が、（ｅ）のテーブルレンダラの定義内容（（Ｂ）の第１６行目）に基づいて生成される。

【０１０１】ここまでで、先に行なった（ｂ）のテーブルレンダラの定義（（Ｂ）の第７行目）におけるレンダラエレメントタグ<uomf:children>についての展開が完了し、そのタグに続く次の定義の記述に基づいて（Ｃ）の生成ＨＴＭＬ文の第１１行目が生成され、テーブルの２行目の表示のためのＨＴＭＬ文の生成が完了する。

【０１０２】（Ｃ）の生成ＨＴＭＬ文の第１２行目から第１６行目まで（テーブルの３行目の表示のためのＨＴＭＬ文）が生成される流れは、上述したテーブルの２行目の表示のためのＨＴＭＬ文（（Ｃ）の生成ＨＴＭＬ文の第７行目から第１１行目まで）と同様である。

【０１０３】ここまでで、先に行なった（ａ）のテーブルレンダラの定義（（Ｂ）の第４行目）におけるレンダラエレメントタグ<uomf:children>についての展開が完了し、そのタグに続く次の定義の記述に基づいて（Ｃ）の生成ＨＴＭＬ文の第１７行目が生成される。こうして、（Ｃ）に示すテーブルの表示のためのＨＴＭＬ文の生成が完了する。

【０１０４】なお、前述したように、生成されたＨＴＭＬ文はクライアントに宛てて送出される。この後、このＨＴＭＬ文に対応するリクエストがクライアントから送付されてきたときにそのリクエストがどのモデルオブジェクトについてのものであるかを容易に特定できるようにするために、ブラウザによって入力フィールドがクライアントで表示されない、いわゆるｈｉｄｄｅｎ属性のｉｎｐｕｔタグ（<input type=hidden>）がこの生成されたＨＴＭＬ文に追加されて送出される。このｉｎｐｕｔタグには、モデルの種類を示す文字列とＨＴＭＬ文の生成に用いられたビュー毎に一意なＩＤ（識別子）とが記述されるようにして、これらのデータがそのリクエストに自動的に含まれるようにする。

【０１０５】図２４に示すテーブルモデル用のフレームワークエンジン３００３が、ブラウザ３００５によってクライアントにテーブルを表示させるためのＨＴＭＬ文を生成する、以上までに説明した処理を、図２７、図２８、及び図２９に示すフローチャートに沿って更に説明する。なお、この処理を「表示時の処理」と称することとする。

【０１０６】フレームワークエンジン３００３は、まず、表示用ＪＳＰ３００４を読み込み、表示用ＪＳＰ３００４に含まれているビュータグを検出する。続いてそのビュータグによって示されているビューに含まれている各レンダラを、レンダラタグで指定されている“type”及び“cls”に基づいて分類してバッファの所定の場所に登録する。更に、この登録を終えた後に、フレームワークエンジン３００３は、ビュータグに指定されているモデルオブジェクト３００１からデータを取得しながら図２７、図２８、及び図２９に示すフローチャートに沿った処理を開始する。なお、ここまでの処理を「ビュータグの解析処理」と総称することとする。

【０１０７】図２７には、テーブルモデル用のフレームワークエンジン３００３によって行なわれる表示時の制御処理の全体フローが示されている。

【０１０８】図２７に示す処理が開始されると、まず、ビューの一意ＩＤが生成される（Ｓ３１０１）。ビューの一意ＩＤは、前述したように、ＨＴＭＬ文の生成に用いられる表示用ＪＳＰ３００４のビュー毎に一意なＩＤであり、例えば“ｕｊｉ．ｍｏｄｅｌ．００００１”などのように、共通のプレフィックス（接頭辞）（“ｕｊｉ．ｍｏｄｅｌ．”の部分）とＨＴＭＬ文生成の度に異なる連番の数字（“００００１”の部分）とを組み合わせたものを生成するようにすればよい。このビュー毎に異なる一意なＩＤは、複数のモデルオブジェクトについての表示がクライアントで行なわれ、それらの各々についてのリクエストをサーバが受け取ったときに、受け取ったリクエストがそれぞれどのモデルオブジェクトにつ

いてのものであるかを識別できるようにするために用いられるものであり、クライアントではモデルオブジェクトに対応するＩＤを含んだリクエストが生成される。

【０１０９】続いて、前述した処理によってフレームワークエンジン３００３の有するバッファに登録されている各レングラから、“type”がtable に指定されているものが取得される（Ｓ３１０２）。

【０１１０】その後、レベルをtable とし、子レベルをrow としたときのレングラ表示処理が実行される（Ｓ３１０３）。レングラ表示処理は図２８にフローチャートで示した処理であり、その詳細は後述する。

【０１１１】上述したＳ３１０３の処理によってテーブルを表示させるためのＨＴＭＬ文が生成される。そこで、フレームワークエンジン３００３は、前述したｈｉｄｄｅｎ属性のｉｎｐｕｔタグを生成してこのＨＴＭＬ文に追加する（Ｓ３１０４）。ここで生成されるｉｎｐｕｔタグは例えば下記のようなものとする。

<input type=hidden name=” uji.model” value=” uji.model.00001” >

<input type=hidden name=” uji.model.00001” value=” table” >

ここで、“ｕｊｉ．ｍｏｄｅｌ．００００１”は先に生成されたビューの一意ＩＤの例である。これらのｉｎｐｕｔタグがＨＴＭＬ文に追加されることによって、このＨＴＭＬ文に対応するリクエストにこれらの情報が含まれるようになり、リクエストとモデルオブジェクト３００１との対応関係が明らかになる。

【０１１２】次に、図２７のＳ３１０３において実行される、図２８にフローチャートで示すレングラ表示処理について説明する。

【０１１３】なお、以降の説明及び図面においては、今までに用いていたレングラエレメントタグの表記（例えば<umof:children>）に加え、uomf: の接頭辞を略した表記（例えば<children>）も併用することとする。

【０１１４】まず、このレングラ表示処理が呼び出されたときのレベルが何であったかが調べられ、このレベルが“type”に指定されているレングラにおける表示方式を定義している要素が未だ残されているか否かが判定され（Ｓ３２０１）、判定結果がＹｅｓならば、次の要素（レングラエレメント）がひとつ取得され（Ｓ３２０２）、Ｓ３２０３に処理が進む。一方、Ｓ３２０１の判定結果がＮｏならばこのレングラ表示処理が終了し、元の処理へ戻る。

【０１１５】続いて、Ｓ３２０２の処理によって取得された要素がどのようなものであるかが判定される（Ｓ３２０３、Ｓ３２０６、Ｓ３２０８、Ｓ３２１０）。

【０１１６】この結果、取得された要素がビュータグであるならば（Ｓ３２０３の判定結果がＹｅｓ）、前述したビュータグの解析処理がこのビュータグに対して行なわれ（Ｓ３２０４）、その後、前述した図２７の処理をこのビュータグについて実行し（Ｓ３２０５）、この処理の終了後はＳ３２０１へ処理が戻る。

【０１１７】このＳ３２０４及びＳ３２０５の処理は、あるビューの記述中に別のビューの記述がなされている（ビューがネストしている）ときに行なわれるものであり、例えば、テーブルモデルのあるセル中に更に別のモデルが存在するような場合に対応させるためのものである。

【０１１８】Ｓ３２０２の処理によって取得された要素が<children>のレングラエレメントタグであるならば（Ｓ３２０６の判定結果がＹｅｓ）、このレングラ表示処理が呼び出されたときの子レベルが何であったかが調べられ、この子レベルについての表示処理が実行され（Ｓ３２０７）、この処理の終了後はＳ３２０１へ処理が戻る。子レベルがrowであるときの表示処理は図２９（Ａ）にフローチャートで示されており、子レベルがcolumnであるときの表示処理は図２９（Ｂ）にフローチャートで示されている。これらの表示処理は後で説明する。

【０１１９】また、Ｓ３２０２の処理によって取得された要素が<name>のレングラエレメントタグであるならば（Ｓ３２０８の判定結果がＹｅｓ）、後にリクエストにおいて使用される要素に名前付けを行なう処理が実行され（Ｓ３２０９）、この処理の終了後はＳ３２０１へ処理が戻る。

【０１２０】ここで、<name>のレングラエレメントタグについて説明する。このタグは、所定の規則に従った名前をフレームワークエンジンに生成させるものである。

【０１２１】本実施例においては、この名前は図３０の（Ａ）に示す規則に従って生成される。ここで、モデル固有位置とは、あるモデルでの自己の位置を一意に表すための文字列であり、例えば、テーブルモデルにおいて２行３列目を表すのであれば、“２＿３”などというように文字列を生成すればよい。このとき、前述したビューの一意ＩＤの生成例（“ｕｊｉ．ｍｏｄｅｌ．００００１”）をそのまま流用すれば、このときに<name>のレングラエレメントタグに応じて生成される名前は図３０の（Ｂ）のようになる。

【０１２２】このような規則による名前付けを行うことにより、この名前からモデルオブジェクト及びそのモデルにおける位置を特定することができる。

【０１２３】<name>のレングラエレメントタグを使用したレングラの定義例を図３１に示す。同図に示す定義の記述を説明すると、<uomf:value/>によってモデルオブジェクトから取得された値が更新前の値としてテーブル中の更新可能なセルに挿入されてクライアントで表示される。ここで、このセルの値がクライアントにおいて更新されるとリクエストが送信される。このリクエストでは、<uomf:name/>で生成される名前が更新後のセルの値を参照するための参照名として用いられる。こうするこ

とによって、テーブルモデルエンジンでは、前述した名前付けの規則により、モデルオブジェクト及びそのモデルにおける位置をこの参照名から特定することができるので、クライアントで生成するひとつのリクエスト中に複数の更新データを含ませるようにすることもでき、更に、異なるモデルオブジェクトについての複数の更新データをひとつのリクエストに含ませることもできる。

【０１２４】また、前述したような、ネストしているビューの中に<uomf:name>のレンダラエレメントタグが記述されている場合には、図３０の（Ｃ）に示す例のように、ネストの外側で生成される名前（例えば図３０の（Ｂ）に示される名前）をプレフィックスとしたものに連番の数字等を組み合わせて前述のビュー一意ＩＤを生成し、そのビュー一意ＩＤにそのネストしているビューにおけるモデル固有位置を組み合わせたものをレンダラエレメントタグについての名前として生成するようにする。この名前付けの規則により、生成された名前から、ネストの存在及びビューの継承関係を認識すること、及びそのネストに係るモデルを特定することができる。

【０１２５】図２８の説明に戻る。

【０１２６】Ｓ３２０２の判定処理によって取得された要素が<value>のレンダラエレメントタグであるならば（Ｓ３２１０の判定結果がＹｅｓ）、このテーブルモデルにおける現在の位置に対応する値がモデルオブジェクトから取得され（Ｓ３２１１）、この後にＳ３２０１へ処理が戻る。

【０１２７】一方、Ｓ３２０２の判定処理によって取得された要素が上述したいずれのタグともことなるものであるならば（Ｓ３２０３、Ｓ３２０６、Ｓ３２０８、Ｓ３２１０の判定結果が全てＮｏ）、この取得された要素が生成ＨＴＭＬ文にそのまま表示され（Ｓ３２１２）、この後にＳ３２０１へ処理が戻る。

【０１２８】以上までの処理がレンダラ表示処理である。

【０１２９】次に、上述したレンダラ表示処理のＳ３２０７において実行される、子レベルについての表示処理を説明する。前述したように子レベルがrowであるときの表示処理は図２９（Ａ）にフローチャートで示されており、子レベルがcolumnであるときの表示処理は図２９（Ｂ）にフローチャートで示されている。

【０１３０】まず、図２９（Ａ）のフローチャートを説明する。

【０１３１】まず、変数ｒｏｗの値が０とされ、この変数ｒｏｗの値がgetRowCount()メソッドの戻り値（すなわちテーブルの行数）を超えるまでＳ３３０２からＳ３３０４までの処理が繰り返される（Ｓ３３０１）。

【０１３２】続いて、モデルオブジェクト３００１から、変数ｒｏｗの値で示される行についての行クラス（getRowClass()）が取得される（Ｓ３３０２）。

【０１３３】次に、フレームワークエンジン３００３の

有するバッファに登録されている各レンダラより、取得された行クラスが“cls ”に指定されており、且つ、“type”がrow に指定されているレンダラが取得される（Ｓ３３０３）。

【０１３４】ここで、レベルをrow とし、子レベルをcolumnとしたときのレンダラ表示処理が実行される（Ｓ３３０４）。ここで実行されるレンダラ処理は、既に説明した、図２８に示されているものである。

【０１３５】その後、変数ｒｏｗの値に１を加算した結果の値が改めて変数ｒｏｗに代入され、Ｓ３３０１へ処理が戻る（Ｓ３３０５）。変数ｒｏｗの値が前述した条件に違したならばこの処理が終了し、元の処理へ戻る。

【０１３６】次に、図２９（Ｂ）のフローチャートを説明する。この処理は、基本的には上述した図２９（Ａ）と同様の処理が実行される。

【０１３７】まず、変数ｃｏｌｕｍｎの値が０とされ、この変数ｃｏｌｕｍｎの値がgetcolumnCount()メソッドの戻り値（すなわちテーブルの列数）を超えるまでＳ３４０２からＳ３４０４までの処理が繰り返される（Ｓ３４０１）。

【０１３８】続いて、モデルオブジェクト３００１から、変数ｒｏｗの値で示される行についての列クラス（getColumnClass()）が取得される（Ｓ３４０２）。

【０１３９】次に、フレームワークエンジン３００３の有するバッファに登録されている各レンダラより、取得された列クラスが“cls ”に指定されており、且つ、“type”がcolumnに指定されているレンダラが取得される（Ｓ３４０３）。

【０１４０】ここで、レベルをcolumnとし、子レベルを無し（ヌル）としたときのレンダラ表示処理が実行される（Ｓ３４０４）。ここで実行されるレンダラ処理も、既に説明した、図２８に示されているものである。

【０１４１】その後、変数ｃｏｌｕｍｎの値に１を加算した結果の値が改めて変数ｃｏｌｕｍｎに代入され、Ｓ３４０１へ処理が戻る（Ｓ３４０５）。変数ｃｏｌｕｍｎの値が前述した条件に達したならばこの処理が終了し、元の処理へ戻る。

【０１４２】以上までの処理が表示処理である。

【０１４３】なお、この表示処理が終了した後に、フレームワークエンジン３００３は、表示用ＪＳＰ３００４のＪＳＰソースとモデルオブジェクト３００１との対応関係が示されている記述（図２６（Ｂ）の例では第１行目）を参照し、図３２に示す記憶領域管理処理を実行する。すなわち、この記述でrequest がtrueに設定されているか否かが判定され（Ｓ３５０１）、この判定結果がＹｅｓならば、フレームワークエンジン３００３の有する記憶部に保持されているモデルオブジェクト３００１の内容をその後も継続し（Ｓ３５０２）、この判定結果がＮｏならば、フレームワークエンジン３００３の有する記憶部におけるモデルオブジェクト３００１の記憶領

域を解放する（Ｓ３５０３）。

【０１４４】次に、図２４に示すシステムにおいて、ブラウザ３００５から発せられるリクエストをフレームワークエンジン３００３が受け取ってモデルオブジェクト３００１を更新する動作について説明する。なお、この動作を「リクエスト時の動作」と称することとする。

【０１４５】図３３は、図２４に示すシステムにおけるリクエスト時の動作の概要を説明する図である。

【０１４６】まず、フレームワークエンジン３００３により生成された、通常の（ｈｉｄｄｅｎ属性でない）<input>タグを含むＨＴＭＬ文をクライアントが受け取り、クライアントに装備されているブラウザ３００５がそのＨＴＭＬ文に基づいてテーブルを表示させている。ここで、この<input>タグに対応するデータがクライアントに入力されると、ブラウザ３００５はこのデータを含むＨＴＴＰによるリクエストをサーバ側に送信する。

【０１４７】サーバのフロントコンポーネント３００６は、このＨＴＴＰリクエストを受信すると、フレームワークエンジン３００３を起動させるための指示を与え、リクエストをフレームワークエンジン３００３に渡す。

【０１４８】フレームワークエンジン３００３では、まず、受け取ったリクエストがどのモデルについてのものかを判断する。この判断は、前述したｈｉｄｄｅｎ属性の<input>タグの作用によってリクエストに含まれるモデルの種類についての情報を利用する。ここで、例えば、このリクエストがテーブルモデルについてのものであると判断されれば、テーブルモデル用のエンジンが選択されて起動される。

【０１４９】続いて、フレームワークエンジン３００３は、同じくｈｉｄｄｅｎ属性の<input>タグの作用によってリクエストに含まれることとなる、ビューの一意ＩＤから、このリクエストに含まれるデータを代入すべきモデルオブジェクト３００１を特定する。

【０１５０】そして、特定されたモデルオブジェクト３００１がフレームワークエンジン３００３内の記憶領域に保持されていればその保持されているものに対してリクエストに含まれていたデータを代入し、保持されていなければ、モデルオブジェクト３００１を改めて生成してそのデータの代入を行なう。

【０１５１】なお、モデルオブジェクト３００１へデータを代入するには、その代入すべきデータをモデルに応じた更新メソッドに変換し、モデルオブジェクトを更新させるインタフェースメソッド（例えば図２５に示したテーブルモデルインタフェース３００２を更新するのであれば、setValueAtメソッド）を呼び出すことにより、モデルオブジェクト３００１の更新を行うようにすればよい。

【０１５２】フレームワークエンジン３００３で行なわれる、上述したリクエスト時の制御処理の処理内容を図３４にフローチャートで示す。

【０１５３】図３４では、この制御処理の全体フローを（Ａ）として示し、この制御処理の途中で実行されるテーブルモデル用のモデルアップデート処理を（Ｂ）として示している。

【０１５４】クライアントのブラウザ３００５で生成された、前述した表示処理によって生成されたＨＴＭＬ文に基づく表示に応じて入力されたデータを含むリクエストには、前述した名前付けの規則に基づくそのデータの参照名が示されていることは既に説明した。そこで、フレームワークエンジン３００３は、リクエストを受け取ると、まず、そのリクエストに含まれているその参照名を全て取り出す。そして、取り出された参照名のうちのひとつが変数ｋｅｙに代入される（Ｓ３６０１）。

【０１５５】ここで、この変数ｋｅｙの値が、フレームワークエンジン３００３で扱えるモデルを表すものであるか否かが判定され（Ｓ３６０２）、この判定結果がＹｅｓならば、続いてその変数ｋｅｙに対応するモデルオブジェクト３００１がフレームワークエンジン３００３の有する記憶部に保持されているか否かが判定される（Ｓ３６０３）。

【０１５６】このＳ３６０３の判定結果がＹｅｓならば、その記憶部に保持されているモデルオブジェクト３００１が取得され（Ｓ３６０４）、このＳ３６０３の判定結果がＮｏならば、改めてモデルオブジェクト３００１が作成される（Ｓ３６０５）。そして、このモデルオブジェクト３００１に対してモデルアップデート処理が施される（Ｓ３６０６）。

【０１５７】モデルアップデート処理を完了した後、若しくはＳ３６０２の判定結果がＮｏであったときには、リクエストから取り出されている別の参照名が変数ｋｅｙに設定され、その後、上述したＳ３６０２からＳ３６０６までの処理が取り出された全ての参照名について行なわれるまで、処理が繰り返される（Ｓ３６０７）。

【０１５８】次に、上述したＳ３６０６の処理において実行される、図３４（Ｂ）に示すテーブルモデル用のモデルアップデート処理について説明する。

【０１５９】まず、変数ｋｅｙの値（すなわち、リクエストに含まれる更新データの参照名）が分解され、その名前を構成している前述したモデル固有位置から、テーブルモデル上の行（row）及び列（column）の位置が取得される（Ｓ３７０１）。

【０１６０】そして、モデルオブジェクト３００１のsetValueAt()メソッドが呼び出され、取得されたテーブルモデルの行と列との位置、及びこの参照名により参照される更新データの値がそのメソッドに渡されることにより、モデルオブジェクト３００１が更新され（Ｓ３７０２）、その後は図３４（Ａ）に処理が戻る。

【０１６１】以上までの処理をフレームワークエンジン３００３が実行することによって、ブラウザ３００５から発せられるリクエストに基づいたモデルオブジェクト

３００１の更新が行なわれる。

【０１６２】以上までに説明した本発明の第３の実施例を実施するクライアントサーバシステムの例を図３５に示す。

【０１６３】サーバ３０１０は、モデルフレームワーク処理部３０１１、Ｗｅｂサーバ部３０１２、バックエンド３０１３を備える。

【０１６４】モデルフレームワーク処理部３０１１は、図２４におけるフレームワークエンジン３００３に相当する機能を実行する。

【０１６５】Ｗｅｂサーバ部３０１２は、モデルフレームワーク処理部３０１１で生成されたＨＴＭＬ文をクライアント（３０２０ａ、３０２０ｂ、３０２０ｃ、…）に送る機能と、クライアント（３０２０ａ、３０２０ｂ、３０２０ｃ、…）からの陰エンストを受け付けてモデルフレームワーク処理部３０１１に渡すフロントコンポーネント３００６に相当する機能とを実行する。

【０１６６】バックエンド３０１３は、データベース３０１４に蓄積されているデータの操作を行ない、モデルオブジェクト３００１を使用してモデルフレームワーク処理部３０１１とデータの授受を行なう。

【０１６７】クライアント（３０２０ａ、３０２０ｂ、３０２０ｃ、…）はブラウザ３００５を装備し、サーバ３０１０から送られてくるＨＴＭＬ文に基づいた画面の表示、及び、その表示画面に含まれている入力フォームへの入力に対応するリクエストの生成及びサーバ３０１０へ宛ての送信を行なうものである。

【０１６８】次に、本発明の更なる実施例について説明する。

【０１６９】これより説明する実施例は、Ｗｅｂアプリケーションを開発する際に、ロジックである処理スクリプトの定義を呼び出す記述を画面の見かけを定義するモジュールには直接記述せずに利用可能とするものであり、処理スクリプトの定義によって表現されるロジックの部品化が可能となり、ロジックと画面定義との分離が進むことによって、ロジックの再利用性を高めてＷｅｂアプリケーションの開発効率を向上させるというものである。

【０１７０】なお、これより説明する上述した実施例を、既に説明した他の実施例と区別するために、「第４の実施例」と称することとする。

【０１７１】まず、図３６について説明する。同図は本発明の第４の実施例によってスクリプトが生成される様子を示しており、（Ａ）に示す記述文に対して後述するコンテンツ変換処理が施されることによって（Ｂ）に示すスクリプトを含むＨＴＭＬ文が生成されることを示している。

【０１７２】図３６（Ａ）及び（Ｂ）の記述内容は、ブラウザによる表示において、ある入力用のフォーム部品に対してなされるクリック操作に応じて、クリック操作

がなされたことを通知する警告画面を表示させる処理を行なう処理スクリプトを実行させるというものである。なお、説明の便宜のため、図３６の（Ａ）及び（Ｂ）の各行頭には行番号を付している。

【０１７３】従来のＷｅｂアプリケーションの開発手法によれば、上述した処理をブラウザに行なわせるためには、図３６（Ｂ）に示すようなＨＴＭＬ文、すなわち、スクリプト部分の呼び出しを行なう条件となるプロパティの記述（図３６（Ｂ）第８行目の記述）と、処理スクリプト自体の動作の定義（図３６（Ｂ）第１行目から第６行目の記述）とを直接記述し、更にこの両者の関連を示すためにその双方に処理スクリプトの名前（図３６（Ｂ）の例ではaction123()）を記述する必要があった。そのため、何らかの理由によってこれらのどちらかに修正が必要となっても、結局両者を参照してその一方に対する修正が他方に影響を与えていないかを確認する必要が生じてしまうため、保守性に問題があった。また、このために、結局は処理スクリプトの動作定義とその処理スクリプトの呼び出しを行なう部分とを同一のコンポーネントに記述することとなり、処理スクリプトの動作定義の再利用性も低かった。

【０１７４】一方、この第４実施例では、同図（Ａ）に示すような各種のタグを用いた記述文を作成し、この記述文に対してコンテンツ変換処理を施すことで（Ｂ）に示すスクリプトを含むＨＴＭＬ文を自動的に生成させるようにするものである。同図（Ａ）には、処理スクリプトに名称が指定されておらず、更に、処理スクリプトを起動させる条件を示すプロパティ（同図（Ｂ）の第８行目の“onclick=”で始まり、処理スクリプト名を指定する記述）も記述されていない。

【０１７５】まず、同図（Ａ）を説明する上で必要な概念を説明する。

【０１７６】図３７は、本発明の第４の実施例で使用されるオブジェクトの構成を示している。

【０１７７】コンポーネント（ScriptComponent）オブジェクト４００１は、ブラウザによって画面中に表示されているオブジェクトに対するイベントの発生元となるオブジェクトであり、図３６（Ａ）の例における<input>タグ（ValidInputTag オブジェクト４０１１）はコンポーネントタグと呼ばれる。コンポーネントタグはアクションタグをその内部に有しており、そのアクションタグに記述されるプロパティの内容である、処理スクリプトの呼び出しを行うためのイベントの指定が行なわれる。

【０１７８】コンテナ（ScriptContainer）オブジェクト４００２は、コンポーネントオブジェクト４００１の管理やアクションオブジェクト４００３の有する処理スクリプトの出力を行なうオブジェクトであり、図３６（Ａ）の例における<form>タグ（ValidFormTagオブジェクト４０１２）はコンテナタグと呼ばれる。なお、コン

テナオブジェクト４００２はコンポーネントオブジェクト４００１を継承しており、自らがイベントの発生元となることもある。

【０１７９】アクション（ScriptAction）オブジェクト４００３は、ロジックである処理の定義がスクリプトによりなされるオブジェクトであり、図３６（Ａ）の例における<action>タグはアクションタグと呼ばれる。なお、アクションタグはカスタムアクションタグ（Custom Actiontag オブジェクト４０１３ａ）とマイアクションタグ（MyActiontagオブジェクト４０１４ｂ）とがあるが、この詳細は後述する。

【０１８０】スクリプト呼び出し部（ScriptCaller）オブジェクト４００４は、イベントの発生に応じて個別に処理スクリプトの呼び出しを行なうときの各処理スクリプトの呼び出し法や戻り値の解釈法を決定するオブジェクトである。

【０１８１】次に、図３８について説明する。同図はコンテンツ変換処理の概要を説明する図である。以下、図３６（Ａ）に示した記述文が変換される様子を図３８を参照しながら説明する。

【０１８２】まず、（Ａ）の第１行目に記述されているコンテナオブジェクト４１０２である<form>タグの記述内容がコンテンツ変換装置を実行する処理装置の有する記憶部に記憶され、第２行目に記述されているコンポーネントオブジェクト４１０１である<input>タグの記述内容が同様に記憶される。

【０１８３】次に、第３行目にアクションオブジェクト４１０３である<action>タグが記述されている。

【０１８４】<action>タグではevent プロパティの指定が必ずなされている。このプロパティの指定によって、このく<action>タグとこれに対応する終了タグとの間に記述されているスクリプトがどのコンポーネントで発生するどのようなイベントに対して実行されるものなのかが示される。（Ａ）の例では、この<action>タグを直接内包している第２行目の<input>タグで発生する“クリック操作”のイベントを対象としていることが示されている。

【０１８５】そこで、コンポーネントオブジェクト４１０１である第２行目の<input> タグに、この<input>が有するアクションオブジェクト４１０３として第３行目の<action>タグが登録される（図３８（ａ））。

【０１８６】次に、第３行目の<action>タグから第５行目の<action>タグの終了タグとの間（すなわち第４行目のみ）に記述されているスクリプト（“alert(“clicked!”)”）がアクションオブジェクト４１０３である<action>タグにおいて実行されるスクリプトとして記憶される。

【０１８７】続いて、第６行目に第２行目の<input> タグに対応する終了タグが記述されている。ここで、<input>タグの生成（すなわち、図３６（Ｂ）の第８行目）が開始され（図３８（ｂ））、<input> タグについての記憶内容（すなわち、図３６（Ａ）の第１行目の記述内容である“inputname=”field1” ”）が出力される。

【０１８８】続いて、この<input> タグに先に登録されていたアクションオブジェクト４１０３である<action>タグに対し出力指示の通知が行なわれる（図３８（ｂ））。

【０１８９】この出力通知が<action>タグによって受け取られると、この<action>タグについての前述したプロパティについての指定内容がコンポーネントオブジェクト４１０１に送られて生成中の<input>タグに追加される（図３８（ｄ））。図３６（Ａ）の例では、event プロパティが“click”に指定されているので、<input>タグには“onclick=”が出力されるようにしておく。ここまでで、コンポーネントオブジェクト４１０１による<input>タグの大枠の生成が完了する。

【０１９０】更に、ここで、前述したスクリプトが保持されているアクションオブジェクト４１０３である<action>タグがコンテナオブジェクト４１０２である<form>タグに登録される（図３８（ｅ））。

【０１９１】最後に、図３６（Ａ）の第７行目に第１行目の<form>タグに対応する終了タグが記述されている。ここで、コンテナオブジェクト４１０２である<form>タグの生成（図３８（ｆ））が開始され、まず、図３６（Ｂ）の第１行目の出力が行なわれる。

【０１９２】続いて、スクリプトの呼び出し部分の生成がスクリプト呼び出し部オブジェクト４１０４に指示される（図３８（ｇ））。ここでは、まず、処理スクリプトの名前（action123(target)）が自動生成されて図３６（Ｂ）の第２行目及び第５行目が出力され、続いてこの処理スクリプトの戻り値が定義される第４行目が出力される。更に、この処理スクリプトの呼び出し元であるコンポーネントオブジェクト４１０１である第８行目の<input>タグに、処理スクリプトの名前を出力する。

【０１９３】そして最後に、<form>タグに先に登録されていたアクションオブジェクト４１０３である<action>タグに指示され（図３８（ｈ））、前述したスクリプトが図３６（Ｂ）の第３行目として出力される。この後に、コンテナオブジェクト４１０２である<form>タグによって、図３６（Ｂ）の第６行目、第７行目、及び第９行目が出力され、図３６（Ｂ）のＨＴＭＬ文の生成が完了する。

【０１９４】以上までに説明した、コンテンツ変換処理をフローチャートで示したものが図３９及び図４０である。次に、このフローチャートについて説明する。

【０１９５】まず、図３９において、処理が開始されると、図３６（Ａ）に示すような記述文が先頭行から読み出される。

【０１９６】この読み出された行にアクションタグが記述されているか否かが判定され（Ｓ４２０１）、アクシ

ョンタグでないならば（Ｓ４２０１の判定結果がＮｏ）、この行に記述されているタグの内容が記憶部に記憶され（Ｓ４２０２）、その後はＳ４２０１へ処理が戻ってこの行の次の行についての判定処理が繰り返される。

【０１９７】一方、読み出された行にアクションタグが記述されているならば（Ｓ４２０１の判定結果がＹｅｓ）、このアクションタグを含んでいるコンポーネントタグにこのアクションタグが登録され（Ｓ４２０３）、続いてこのアクションタグの開始タグと終了タグとの間に記述されている処理スクリプトが記憶部に記憶される（Ｓ４２０４）。

【０１９８】次に、アクションタグの終了タグが記述されている行の次の行が読み出され、この行にコンポーネントタグの終了タグが記述されているか否かが判定される（Ｓ４２０５）。この結果、コンポーネントタグの終了タグがこの行に記述されていなければ（Ｓ４２０５の判定結果がＮｏ）、Ｓ４２０１へ処理が戻ってこの行についての判定処理が繰り返される。

【０１９９】一方、この行にコンポーネントタグの終了タグがこの行に記述されていれば（Ｓ４２０５の判定結果かＹｅｓ）、このコンポーネントタグについての生成が開始されて記憶部に記憶されているこのコンポーネントタグについての記述内容が出力され（Ｓ４２０６）、更に、このコンポーネントタグに登録されているアクションタグのプロパティが追加される（Ｓ４２０７）。

【０２００】次に、図４０に処理が進み、ここで、追加されたアクションタグのプロパティの内容が調べられ、このプロパティで指定されているイベントが、このコンポーネントでサポートされているものであるか否かが判定される（Ｓ４２０８）。この結果、指定されているイベントがこのコンポーネントでサポートされていないものであるならば（Ｓ４２０８の判定結果がＮｏ）、このコンテンツ変換処理の処理結果として元の記述文に誤りが存在することを示すエラー通知が出力され（Ｓ４２０９）、処理が終了する。

【０２０１】一方、指定されているイベントがこのコンポーネントでサポートされているものであったならば（Ｓ４２０８の判定結果がＹｅｓ）、コンポーネントタグの生成が完了し（Ｓ４２１０）、生成されたコンポーネントタグが記憶部に一時的に保管される。更に、このコンポーネントタグに登録されていたアクションタグがコンテナタグに登録される。

【０２０２】次に、コンポーネントタグの終了タグが記述されていた行の次の行が読み出され、この行にコンテナタグの終了タグが記述されているか否かが判定される（Ｓ４２１２）。この結果、コンテナタグの終了タグがこの行に記述されていなければ（Ｓ４２１２の判定結果がＮｏ）、Ｓ４２０１（図３９）へ処理が戻ってこの行についての判定処理が繰り返される。

【０２０３】一方、この行にコンテナタグの終了タグがこの行に記述されていれば（Ｓ４２１２の判定結果かＹｅｓ）、このコンテナタグについての出力が開始されて記憶部に記憶されているこのコンテナタグについての記述内容が出力される（Ｓ４２１３）。

【０２０４】続いて、スクリプト呼び出し部オブジェクトによって処理スクリプトの名前が自動生成されて出力され（Ｓ４２１４）、更に一時的に保管されているコンポーネントタグにその名前が記述されることによってコンポーネントタグと処理スクリプトとが関連付けられる。

【０２０５】その後、このコンテナタグに登録されているアクションタグに関連付けて記述されていて先に記憶部に記憶させていた処理スクリプトが出力され（Ｓ４２１５）、そして最後に<script>タグの出力やコンテナタグの終了タグの出力などが行なわれてコンテナタグの生成が完了する（Ｓ４２１６）。

【０２０６】以上の処理が実行されることによって、図３６（Ａ）に示すような記述文から図３６（Ｂ）に示すようなＨＴＭＬ文が生成される。

【０２０７】なお、このコンテンツ変換処理は、サーバに単独で設けられる処理エンジンによって処理させるようにしてもよいが、前述した本発明の第３の実施例で説明したフレームワークエンジンによって、前述した表示処理に併せて処理させるようにしてもよい。このとき、コンテンツ変換処理での変換対象であるＪＳＰソースは、モデルを表示するための画面の見かけの定義が行なわれるビューにおけるレンダラの要素として記述される。

【０２０８】次にこの第４の実施例の応用について説明する。

【０２０９】図３６では処理を定義するスクリプトを記述文（Ａ）に記述（第４行目）に記述していたが、この処理スクリプトを記述文（Ａ）中では記述せずに、部品化されている処理スクリプトを利用する手法についてまず説明する。

【０２１０】図４１は、処理スクリプトを別に用意する場合のスクリプト記述例を示す図である。

【０２１１】図３６（Ａ）に示した記述文では、全てのタグにsf:なる接頭辞を付していたが、図４１（Ａ）に示す記述文では、アクションタグ<action>にmy:なる接頭辞が付されている。このタグ<my:action>はマイアクションタグと称されており、コンテンツ変換処理においてこのタグが検出されたときには、予め用意されている処理スクリプトがＳ４２１５（図４０）の処理で出力されるようにする。なお、これに対して、今まで説明したsf:なる接頭辞を付したアクションタグは特にカスタムアクションタグと称されている。

【０２１２】図４１（Ａ）に記述されたマイアクションタグに対応するスクリプトである、文字列の最小文字数

をチェックする処理スクリプトの定義例を図４２に示す。図４２に示す定義例では、マイアクションタグに指定されるMinLength プロパティを受け取るためのsetMin Length()が定義され、更に、outputFunctionBodyメソッドにおいて、出力オブジェクトWriterに対して、上述したプロパティの内容を埋め込んだ処理スクリプトを出力させることが指示されている。

【０２１３】図３９及び図４０に示されているコンテンツ変換処理を実行させると、図３９のＳ４２０４の処理において図４２に示されているような処理スクリプトが記憶部に記憶され、その後の図４０のＳ４２１５の処理において、その処理スクリプトが出力されるときに、変数min への値の代入がなされた出力が行なわれる。その結果、図４１（Ａ）の記述文から図４１（Ｂ）に示すＨＴＭＬ文が生成される。

【０２１４】次に図４３について説明する。同図は、同一のイベントに対して複数のアクションが対応する場合のスクリプト記述例を示す図である。

【０２１５】図４３（Ａ）の記述文では、第２行目に示されている<input〉タグが、第３行目及び第６行目の２つのアクションタグを有している。このような場合には、図３９及び図４０に示されているコンテンツ変換処理において、図３９のＳ４２０５の判定処理の作用により、Ｓ４２０１からＳ４２０５にかけての処理が２度行なわれ、その結果、〈input〉タグにこれら２つのアクションタグが登録され、更にそれぞれのアクションタグについての処理スクリプトが記憶される。

【０２１６】その後、図３９のＳ４２０７の処理において２つのアクションタグから<input〉タグにプロパティが追加されると、コンテンツ変換処理では、これら２つのアクションタグに指定されている”event” の内容が同一であることが認識される。

【０２１７】この認識結果に基づいて、コンテンツ変換処理では、図４０のＳ４２１４の処理において、処理スクリプトの名前の自動生成に加え、図４３（ｂ）の第２行目から第８行目（（ａ）の部分）に示す、２つの処理スクリプトを順に実行させるための処理スクリプトの自動生成がスクリプト呼び出し部オブジェクトで行なわれる。このスクリプト呼び出し部オブジェクトによる処理スクリプトの自動生成によって、同一イベントと複数の処理スクリプトとの対応関係が確立される。

【０２１８】その後、Ｓ４２１５において、図４３（Ｂ）の第９行目から第１２行目（（ｂ）の部分）及び第１３行目から第１９行目（（ｃ）の部分）の出力が行なわれる。

【０２１９】次に図４４について説明する。同図は、コンテナタグでイベントが発生する場合のスクリプト記述例を示している。

【０２２０】図４４（Ａ）において、第６行目のアクションタグの記述において、event プロパティが”” for m.submit” ”に指定されている。この記述は、mameプロパティが”form”に指定されているコンテナタグについてのアクションタグであることを示している。

【０２２１】このようなプロパティの指定がなされているコンテナタグについては、図３９及び図４０に示されているコンテンツ変換処理において、図３９のＳ４２０３の処理であるアクションタグの登録が、コンポーネントタグの代わりにコンテナタグに対して行なわれるようにする。更に、図４０のＳ４２１３の処理におけるコンテナタグについての記述内容の出力の際に、Ｓ４２０７からＳ４２１０にかけてのコンポーネントタグに対して行なわれる処理がコンテナタグに対して施されるようにする。こうすることにより、図４４（Ｂ）の第１４行目（（ａ）の行）が生成されるようになり、コンテナタグで発生するイベントに対してスクリプトを実行させることが可能となる。

【０２２２】以上まで説明した本発明の各実施例を実施するシステムで使用される、サーバの各処理エンジン及びクライアントの構成を図４５に示す。これらはいずれも、ＣＰＵ５００１、記憶部５００２、入力部５００３、出力部５００４、Ｉ／Ｆ部５００５を有し、各構成要素がバス５００６を介して相互に接続されている。

【０２２３】各構成要素の機能を説明すると、ＣＰＵ（中央処理装置）５００１は制御プログラムを実行することで各構成要素を制御する。

【０２２４】記憶部５００２はＲＯＭ（リードオンリメモリ）やＲＡＭ（ランダムアクセスメモリ）、磁気記憶装置等を備えており、ＣＰＵ５００１に各構成要素を制御させる制御プログラムの記憶、ＣＰＵ５００１が制御プログラムを実行する際のワークエリア、あるいは各種データの記憶領域として使用される。

【０２２５】入力部５００３はマウスやキーボード等を有しており、ユーザによる操作に対応する各種のデータが取得される。

【０２２６】出力部５００４はディスプレイなどを有しており、各種のデータを提示してユーザに通知するものである。

【０２２７】Ｉ／Ｆ部５００５はネットワークに接続するためのインタフェース機能を提供するものであり、ネットワークを介して他の機器とのデータ授受を可能とするものである。

【０２２８】なお、この図４５に示されている構成は、標準的なコンピュータが有しているものと同様の構成であり、従って、本発明を標準的なコンピュータで実施することも勿論可能である。

【０２２９】図４６は、本発明に係わるコンピュータ等の情報処理装置で実行される各種のソフトウェアプログラム等の提供方法を説明する図である。プログラム等は例えば以下の（ａ）～（ｃ）の３つの方法の中の任意の方法により提供される。

【０２３０】（ａ）コンピュータ等の情報処理装置５３０１にインストールされて提供される。この場合、プログラム等は例えば出荷前にプレインストールされる。

【０２３１】（ｂ）可搬型記録媒体５３０２に格納されて提供される。この場合、可搬型記憶媒体５３０２に格納されているプログラム等は、コンピュータ等の情報処理装置２３０１の外部記憶装置にインストールされる。可搬型記憶媒体５３０２の例としては、フロッピー（登録商標）ディスク、ＣＤ－ＲＯＭ、光磁気ディスク、ＤＶＤ－ＲＯＭなどがある。

【０２３２】（ｃ）ネットワーク５３０３上のプログラム提供サーバ５３０４から提供される。この場合、基本的には、コンピュータなどの情報処理装置５３０１がプログラム提供サーバ５３０４に格納されているプログラム等をダウンロードすることによって、そのプログラム等を取得する。この場合には、ソフトウェアプログラムを表現するデータ信号で搬送波を変調して得られる伝送信号を、プログラム提供サーバ５３０４から伝送媒体であるネットワーク５３０３を通じて伝送し、情報処理装置５３０１では受信した伝送信号を復調してソフトウェアプログラムを再生することで当該ソフトウェアプログラムの実行が可能となる。

【０２３３】（付記１）　クライアントからのリクエストをサーバで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、前記サーバで、前記リクエストをデータオブジェクトに変換して処理し、処理結果であるデータオブジェクトを前記クライアントへのレスポンスに変換して返すことを特徴とするＷｅｂシステム。

【０２３４】（付記２）　Ｗｅｂアプリケーションを開発・実行するためのシステムであって、入力内容をデータオブジェクトに変換する入力内容変換手段と、複数の処理ルーチンを備える処理ロジックと、前記データオブジェクトの種類とコマンドの組合せを前記各処理ルーチンにマッピングする第１の外部定義ファイルと、前記データオブジェクトの種類と前記コマンドと前記第１の外部定義ファイルに基づいて前記処理ロジックの備える処理ルーチンから適当な処理ルーチンを決定する処理ルーチン決定手段と、を備えることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２３５】（付記３）　付記２記載のＷｅｂアプリケーション開発・実行システムであって、前記入力内容変換手段は、ＨＴＭＬで記述されたデータ入力ページの特定項目を前記データオブジェクトのクラス名とし、該データ入力用ページに設けられている各入力欄の名前を前記データオブジェクトの属性に対応させて、データオブジェクト用プログラムを自動生成することを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２３６】（付記４）　Ｗｅｂアプリケーションを開発・実行するためのシステムであって、複数の処理ルーチンを備える処理ロジックと、前記処理ロジックの処理結果とデータオブジェクトの種類の組み合わせを表示用コンポーネントにマッピングする第２の外部定義ファイルと、を備えることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２３７】（付記５）　付記４記載のＷｅｂアプリケーション開発・実行システムであって、更に、表示するページに配置する複数の表示用コンポーネントの配置の仕方を規定したテンプレートファイルを備え、前記テンプレートファイルに基づいて複数の前記処理ロジックの処理結果を出力することを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２３８】（付記６）　付記２記載のＷｅｂアプリケーション開発・実行システムであって、更に、ＸＭＬのタグとデータオブジェクトをマッピングするＸＭＬマッピングファイルと、ＸＭＬのタグとデータオブジェクトとの相互変換を行うＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンを備え、前記入力内容としてＸＭＬで記述されたタグを受信した場合に、前記ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンは、前記受信したＸＭＬのタグと前記ＸＭＬマッピングファイルに基づいて、前記受信したＸＭＬを前記データオブジェクトに変換し、前記処理ルーチン決定手段は、前記データオブジェクトと前記受信したＸＭＬのタグと前記第１の外部定義ファイルに基づいて前記処理ロジック内の処理ルーチンから適当な処理ルーチンを決定し、前記ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジンは、前記処理ロジックの処理結果として得られるデータオブジェクトをＸＭＬのタグに変換して出力する、ことを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２３９】（付記７）　付記６記載のＷｅｂアプリケーション開発・実行システムであって、前記ＸＭＬマッピングファイルは、或るＨＴＴＰによるデータと同一の処理を施すＸＭＬのタグを、前記或るＨＴＴＰによるデータと同一種類のデータオブジェクトにマッピングすることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４０】（付記８）　付記２記載のＷｅｂアプリケーション開発・実行システムであって、前記処理ロジックにおける各処理ルーチンは、時間的スコープの大きな順にシステム、アプリケーション、セッション、リクエストの４段階のうちのいずれかの前記スコープに属性定義されており、前記処理ロジックが処理を行う場合には、より大きなスコープからより小さなスコープに分岐して処理を進めていくことを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４１】（付記９）　付記８記載のＷｅｂアプリケーション開発・実行システムであって、いずれかの前記スコープが属性定義されている前記処理ルーチンに前処理または後処理のインタフェースを設けることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４２】（付記１０）　付記９記載のＷｅｂアプリケーション開発・実行システムであって、前記前処理のインタフェースに状態の制御を行うためのチェックルーチンを設けることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４３】（付記１１）　付記９記載のＷｅｂアプリケーション開発・実行システムであって、前記後処理のインタフェースに前記スコープが属性定義されている前記処理ルーチンごとのエラーを回復させるためのエラーハンドリング処理を設けることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４４】（付記１２）　付記８記載のＷｅｂアプリケーション開発・実行システムであって、前記スコープが属性定義されている前記処理ルーチンのいずれかにシングルスレッド動作制限を設けることを特徴とするＷｅｂアプリケーション開発・実行システム。

【０２４５】（付記１３）　コンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、入力内容をデータオブジェクトに変換するステップと、前記データオブジェクトの種類と、コマンドと、前記データオブジェクトの種類とコマンドの組合せを各処理ルーチンにマッピングする外部定義ファイルと、に基づいて処理ロジック内の処理ルーチンから適当な処理ルーチンを決定するステップと、を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読み出し可能記録媒体。

【０２４６】（付記１４）　クライアントからのリクエストをサーバで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、前記サーバで前記リクエストに応じて処理される各オブジェクトは、各々が処理されるときの時間軸上における相対的な位置関係に基づいた時間的スコープについての属性定義がされており、前記サーバは、より大きな時間的スコープが定義されているオブジェクトから小さな時間的スコープが定義されているオブジェクトに分岐して前記リクエストに応じたオブジェクトの処理を進めていくことを特徴とするＷｅｂシステム。

【０２４７】（付記１５）　付記１４記載のＷｅｂシステムであって、各オブジェクトには、時間的スコープの大きな順に、システム、アプリケーション、セッション、リクエストのうちのいずれかのスコープの属性定義がなされることを特徴とするＷｅｂシステム。

【０２４８】（付記１６）　クライアントからのリクエストをサーバで処理し、クライアントにレスポンスを返すＷｅｂシステムであって、サーバでの処理ルーチン呼び出し時に該処理ルーチンによる処理の動作を監視し、該処理の進行を継続させるオブジェクトを該サーバで実行することを特徴とするＷｅｂシステム。

【０２４９】（付記１７）　クライアントとの間で各種のデータを授受するサーバ側に設けられ、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成装置であって、前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトが格納されるデータオブジェクト格納手段と、前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文が格納される定義文格納手段と、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＨＴＭＬ文生成手段と、を有することを特徴とするＷｅｂアプリケーション生成装置。

【０２５０】（付記１８）　付記１７記載のＷｅｂアプリケーション生成装置であって、前記定義文はＪａｖａ　Ｓｅｒｖｅｒ　Ｐａｇｅｓ（ＪＳＰ）によって定義され、前記ＨＴＭＬ文生成手段は、ＪＳＰによって定義された定義文から前記ＨＴＭＬ文を生成する、ことを特徴とするＷｅｂアプリケーション生成装置。

【０２５１】（付記１９）　付記１７記載のＷｅｂアプリケーション生成装置であって、前記データ構造に対応して定義される属性であるデータクラスを取得するデータクラス取得手段を更に有し、前記ＨＴＭＬ文生成手段は、前記データクラスに基づいて前記データオブジェクトの有するデータに関連付ける前記定義文を選択する、ことを特徴とするＷｅｂアプリケーション生成装置。

【０２５２】（付記２０）　付記１７記載のＷｅｂアプリケーション生成装置であって、前記ＨＴＭＬ文生成手段により生成されたＨＴＭＬ文によって表現されるＷｅｂページ画面に示されている入力フォームへの入力データを含むリクエストであって前記クライアントから送付される該リクエストを取得するリクエスト取得手段と、前記リクエストに前記データと共に含まれている、前記ＨＴＭＬ文生成手段により生成されたＨＴＭＬ文に含まれている文字列であって、該ＨＴＭＬ文の生成の基礎とされたインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる該文字列に基づいて、該文字列で特定されるインタフェースにより表されるデータ構造における特定の位置のデータを該リクエストに含まれているデータに更新するデータ更新手段と、を更に有することを特徴とするＷｅｂアプリケーション生成装置。

【０２５３】（付記２１）　付記２０記載のＷｅｂアプリケーション生成装置であって、前記リクエストに前記データと共に含まれている文字列は、該入力フォームに対して入力されるデータを参照するパラメータ名を示すことを特徴とするＷｅｂアプリケーション生成装置。

【０２５４】（付記２２）　付記２０記載のＷｅｂアプ

リケーション生成装置であって、前記リクエストには、異なるインタフェースに基づいて生成されたＨＴＭＬ文についての前記データ及び前記文字列が含まれることを特徴とするＷｅｂアプリケーション生成装置。

【０２５５】（付記２３） 付記２０記載のＷｅｂアプリケーション生成装置であって、前記データ更新手段は、前記データオブジェクト格納手段に格納されているデータオブジェクトの有する前記データを前記リクエストに含まれていたデータに更新することを特徴とするＷｅｂアプリケーション生成装置。

【０２５６】（付記２４） 付記２０記載のＷｅｂアプリケーション生成装置であって、前記リクエストに前記データと共に含まれている文字列は、前記ＨＴＭＬ文の生成の基礎とされたインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる文字列に、更に該位置のデータ群が有するデータ構造を表すインタフェースを特定する文字列と該インタフェースにより表されるデータ構造における特定の位置を示す文字列とからなる文字列を組み合わせて成り、前記データ更新手段は、前記文字列で特定されるデータ構造における特定の位置に更に有しているデータ構造における特定の位置のデータを前記リクエストに含まれていたデータに更新する、ことを特徴とするＷｅｂアプリケーション生成装置。

【０２５７】（付記２５） クライアントとの間で各種のデータを授受するサーバ側で、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する方法であって、前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得し、前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得し、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する、を有することを特徴とするＷｅｂアプリケーション生成方法。

【０２５８】（付記２６） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを記録した記録媒体であって、前記クライアントに提供するデータを有し、且つ該データのデータ構造を表す，インタフェースを実装するデータオブジェクトを取得する制御と、前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する制御

と、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御と、をコンピュータに行なわせる制御プログラムを記憶した記憶媒体。

【０２５９】（付記２７） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを含む搬送波に具現化されたコンピュータ・データ・シグナルであって、該制御プログラムは、前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得する制御と、前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する制御と、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御と、をコンピュータに行なわせる。

【０２６０】（付記２８） クライアントとの間で各種のデータを授受するサーバ側に設けられ、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成装置であって、処理の内容が定義されている処理ロジックが格納される処理ロジック格納手段と、前記処理ロジックの実行条件が格納される実行条件格納手段と、前記処理ロジックの名称となる文字列を生成する処理ロジック名生成手段と、前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成するＨＴＭＬ文生成手段と、を有することを特徴とするＷｅｂアプリケーション生成装置。

【０２６１】（付記２９） 付記２８記載のＷｅｂアプリケーション生成装置であって、前記処理ロジックはスクリプトによって定義されていることを特徴とするＷｅｂアプリケーション生成装置。

【０２６２】（付記３０） 付記２８記載のＷｅｂアプリケーション生成装置であって、前記処理ロジック格納手段及び前記実行条件格納手段には、前記処理ロジック及び前記実行条件が共に定義されている同一のコンポーネントにおける該処理ロジック及び該実行条件がそれぞれ格納されることを特徴とするＷｅｂアプリケーション生成装置。

【０２６３】（付記３１） 付記２８記載のＷｅｂアプリケーション生成装置であって、前記処理ロジック格納手段には複数の前記処理ロジックが格納され、複数の前記処理ロジックのそれぞれの前記実行条件のうちのいずれかが同一であるときに、該実行条件が同一である該処理ロジックを順に実行する処理ロジックを生成する処理ロジック生成手段を更に有し、前記文字列生成手段は、複数の前記処理ロジック、及び前記処理ロジック生成手段により生成された処理ロジックに対して各々異なる名称となる文字列を生成し、前記ＨＴＭＬ文生成手段は、前記文字列を用いて前記処理ロジック生成手段により生成された処理ロジックを呼び出すＨＴＭＬ文であって、同一であった前記実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する、ことを特徴とするＷｅｂアプリケーション生成装置。

【０２６４】（付記３２） 付記２８記載のＷｅｂアプリケーション生成装置であって、前記処理ロジックと該処理ロジックの実行条件との対応関係の妥当性を確認する確認手段を更に有することを特徴とするＷｅｂアプリケーション生成装置。

【０２６５】（付記３３） クライアントとの間で各種のデータを授受するサーバ側で、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成するＷｅｂアプリケーション生成方法であって、処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成し、前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する、ことを特徴とするＷｅｂアプリケーション生成方法。

【０２６６】（付記３４） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを記録した記録媒体であって、処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する制御と、前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する制御と、をコンピュータに行なわせる制御プログラムを記憶した記憶媒体。

【０２６７】（付記３５） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムを含む搬送波に具現化されたコンピュータ・データ・シグナルであって、該制御プログラムは、処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する制御と、前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する制御と、をコンピュータに行なわせる。

【０２６８】（付記３６） コンピュータに、入力内容をデータオブジェクトに変換するステップと、前記データオブジェクトの種類と、コマンドと、前記データオブジェクトの種類とコマンドの組合せを各処理ルーチンにマッピングする外部定義ファイルと、に基づいて処理ロジック内の処理ルーチンから適当な処理ルーチンを決定するステップと、を実行させるためのプログラム。

【０２６９】（付記３７） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムであって、前記クライアントに提供するデータを有し、且つ該データのデータ構造を表すインタフェースを実装するデータオブジェクトを取得する制御と、前記データオブジェクトが有するデータについての前記Ｗｅｂページ画面における表示方法をＨＴＭＬによる記述によって定義する定義文を取得する制御と、前記データオブジェクトに実装されている前記インタフェースに基づいて該データオブジェクトの有するデータと前記定義文とを関連付けることによって、該データが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御と、をコンピュータに行なわせるための制御プログラム。

【０２７０】（付記３８） クライアントとの間で各種のデータを授受するサーバ側で、コンピュータに実行させることによって、該クライアントに提供するデータが表示されるＷｅｂページ画面を表現するＨＴＭＬ文を生成する制御を該コンピュータに行なわせる制御プログラムであって、処理の内容が定義されている処理ロジックであって、予め用意されている該処理ロジックの名称となる文字列を生成する制御と、前記文字列を用いて前記処理ロジックを呼び出すＨＴＭＬ文であって、前記処理ロジックについての実行条件であって予め用意されている該実行条件に合致するイベントが前記クライアントで

発生したときに該処理ロジックを呼び出して実行する、という処理を該クライアントに行なわせる該ＨＴＭＬ文を生成する制御と、をコンピュータに行なわせるための制御プログラム。

【０２７１】

【０２７２】

【発明の効果】以上詳細に説明したように、本発明によれば、Ｗｅｂアプリケーションシステムにおいて画面表示を規定するＨＴＭＬ、データオブジェクト、処理の内容を規定するロジックの連携が疎となることになり、各モジュールの再利用性を向上させ、開発効率と保守性を上げることができる。また、サーバにおける処理を時間的スコープの大きなものから小さなものに分岐して処理を進めることにより、スレッドセーフを実現したり、各スコープにおける処理に前処理、後処理を追加し、処理の前判断やエラー処理を柔軟に行えるという効果を得ることができる。

【図面の簡単な説明】

【図１】本発明の概略を示す図である。

【図２】本発明の原理構成を示す図である。

【図３】本発明の第１の実施例のシステム構成を示す図である。

【図４】本発明の第１の実施例のシステムの具体的な動作の様子を示す図である。

【図５】図４に示すアプリケーションのシステム構成とその動作（クライアントからのリクエストを受信し、処理するまで）概要を説明する図である。

【図６】図４に示すアプリケーションのシステム構成とその動作（クライアントへレスポンスを返すまで）概要を説明する図である。

【図７】（ａ）、（ｂ）はコマンドマッピングの一部、（ｃ）はページマッピングの一部を示す図である。

【図８】クライアントからのリクエストデータを入力用Ｊａｖａ　Ｂｅａｎに変換するプログラム記述を示す図である。

【図９】コマンドマッピングを介して実行すべきロジックの処理を決定するプログラム記述を示す図である。

【図１０】ユーザロジックが表示用データオブジェクトを設定するプログラム記述を示す図である。

【図１１】ページマッピングを介して表示すべきページを決定するプログラム記述を示す図である。

【図１２】ＪＳＰから表示画面を出力するプログラム記述を示す図である。

【図１３】表示画面に複数の表示部品を配置する場合のテンプレートについて説明する図である。

【図１４】本発明の第２の実施例のシステム構成を示す図（その１）である。

【図１５】本発明の第２の実施例のシステム構成を示す図（その２）である。

【図１６】ＸＭＬファイルを処理するシステムの動作（ＸＭＬファイルを受信して、処理するまで）概要を示す図である。

【図１７】ＸＭＬファイルを処理するシステムの動作（ＸＭＬファイルを受信した時に、処理結果を返すまで）概要を示す図である。

【図１８】（ａ）は、本発明のシステムを構成するオブジェクトのスコープを説明する図であり、（ｂ）はオブジェクト間の相関関係を示す図である。

【図１９】（ａ）は、クライアントからのリクエスト処理を説明する図であり、（ｂ）はSingleThreadModelの実装を説明する図である。

【図２０】（ａ）は、SingleThreadModel の具体的な実装を示す図であり、（ｂ）はSingleThreadModelの実装のプログラム記述例を示す図である。

【図２１】クライアントのリクエストを処理する場合のシーケンス図である。

【図２２】（ａ）は、一般のアプリケーションサーバにおいてロジックのハンドラからブラウザにコールバックする仕組みを説明する図であり、（ｂ）はそのプログラム記述例を示す図である。

【図２３】（ａ）は、本発明のアプリケーションサーバにおいてロジックのハンドラからブラウザにリクエストを行う仕組みを説明する図であり、（ｂ）、（ｃ）はそのプログラム記述例を示す図である。

【図２４】本発明の第３の実施例のシステム構成を示す図である。

【図２５】テーブルモデルインタフェースの宣言例を示す図である。

【図２６】本発明の第３の実施例において、ＨＴＭＬ文が生成される様子を説明する図である。

【図２７】表示時の制御処理の処理内容を示すフローチャート（その１）である。

【図２８】表示時の制御処理の処理内容を示すフローチャート（その２）である。

【図２９】表示時の制御処理の処理内容を示すフローチャート（その３）である。

【図３０】<name>のレンダラエレメントタグを説明する図である。

【図３１】<name>のレンダラエレメントタグを使用したレンダラの定義例を示す図である。

【図３２】記憶領域管理処理の処理内容を示すフローチャートである。

【図３３】図２４に示すシステムにおけるリクエスト時の動作の概要を説明する図である。

【図３４】リクエスト時の制御処理の処理内容を示すフローチャートである。

【図３５】本発明の第３の実施例を実施するクライアントサーバシステムの例を示す図である。

【図３６】本発明の第４の実施例によってＨＴＭＬ文が生成される様子を示す図である。

【図３７】本発明の第４の実施例で使用されるオブジェクトの構成を示す図である。
【図３８】コンテンツ変換処理の概要を説明する図である。
【図３９】コンテンツ変換処理の処理内容を示すフローチャート（その１）である。
【図４０】コンテンツ変換処理の処理内容を示すフローチャート（その２）である。
【図４１】処理スクリプトを別に用意する場合のスクリプト記述例を示す図である。
【図４２】文字列の最小文字数をチェックする処理スクリプトの定義例を示す図である。
【図４３】同一のイベントに対して複数のアクションが対応する場合のスクリプト記述例を示す図である。
【図４４】コンテナタグでイベントが発生する場合のスクリプト記述例を示す図である。
【図４５】本発明の各実施例を実施するシステムに使用される、サーバの各処理エンジン及びクライアントの構成を示す図である。
【図４６】本発明に係わるソフトウェアプログラム等の提供方法を説明する図である。
【図４７】ビジネスアプリケーションの実施形態を示す図である。
【図４８】Ｓｅｒｖｌｅｔを用いたアプリケーションサーバのシステム構成を示す図である。
【図４９】ＪＳＰを用いたアプリケーションサーバのシステム構成を示す図である。
【符号の説明】
１０１　ＨＴＭＬ
１０２　画面データ
１０３　ロジック
２０１　入力用ＪＳＰ
２０２　表示用ＪＳＰ
２０３　入力用Ｊａｖａ　Ｂｅａｎ
２０４　表示用Ｊａｖａ　Ｂｅａｎ
２０５　ユーザロジック
２０６　ＵＪＩエンジン
２０７　マッピングファイル
３０１　ＨＴＴＰリクエスト
３０２　フロントコンポーネント
３０３　入力用Ｊａｖａ　Ｂｅａｎ
３０４　ＵＪＩエンジン
３０５　コマンドマッピング
３０６　ページマッピング
３０７　ユーザロジック
３０８　テンプレート
３０９　表示用Ｊａｖａ　Ｂｅａｎ
３１０　表示用ＪＳＰ
４０１　ユーザー一覧表示画面
４０２　ユーザ状態編集画面
４０３　プロファイル編集画面
４０４　ユーザ登録画面
４０５　ログインボタン
４０６　ユーザ登録ボタン
４０７　変更ボタン
４０８　ログアウトボタン
４０９　プロファイル編集ボタン
４１０　変更ボタン
４１１　戻るボタン
４１２　変更ボタン
４１３　戻るボタン
５０１　ユーザー一覧表示画面
５０２　ユーザ状態編集画面
５０３　プロファイル編集画面
５０４　ユーザ登録画面
５０５　Ｊａｖａ　Ｂｅａｎ（自動生成）
５０６　ＵＪＩ
５０７　コマンドマッピング
５０８　ログイン処理
５０９　ログアウト処理
５１０　ユーザ状態変更処理
５１１　プロファイル変更処理
５１２　ユーザ登録処理
６０１　Ｊａｖａ　Ｂｅａｎ（処理内で作成）
６０２　ページマッピング
８０１　データ入力用ＨＴＭＬページ
８０２　入力用Ｊａｖａ　Ｂｅａｎ
９０１　データ入力用ＨＴＭＬページ
９０２　コマンドマッピング
９０３　ユーザロジック（ハンドラ）
１００１　ユーザロジック（ハンドラ）
１００２　データベース
１００３　表示用Ｊａｖａ　Ｂｅａｎ
１１０１　表示用Ｊａｖａ　Ｂｅａｎ
１１０２　ページマッピング
１１０３　login-succeded.jsp
１１０４　login-failed.jsp
１２０１　表示用ＪＳＰ
１２０２　表示用Ｊａｖａ　Ｂｅａｎ
１２０３　出力ＨＴＭＬ
１３０１　ユーザロジック（ハンドラ）
１３０２　表示用Ｊａｖａ　Ｂｅａｎ（バナー用）
１３０３　表示用Ｊａｖａ　Ｂｅａｎ（メニュー用）
１３０４　表示用Ｊａｖａ　Ｂｅａｎ（コンテンツ用）
１３０５　テンプレート
１３０６　出力用ＨＴＭＬ
１４０１　ＸＭＬインスタンス
１４０２　コマンドマッピング
１４０３　ＵＪＩエンジン
１４０４　ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジン

53

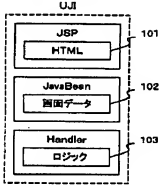
１４０５　Ｂｅａｎインスタンス
１５０１　ロジック（ハンドラ）
１５０２　送信用のデータオブジェクト
１５０３　送信ＸＭＬインスタンス
１６０１　注文伝票ＸＭＬ
１６０２　出荷伝票ＸＭＬ
１６０３　ＵＪＩ
１６０４　ＸＭＬ　Ｄａｔａ　Ｂｉｎｄｉｎｇエンジン
１６０５　Ｊａｖａ　Ｂｅａｎ
１６０６　ＸＭＬマップファイル
１６０７　コマンドマッピング
１６０８　注文処理
１６０９　出荷処理
１６１０　処理
１７０１　Ｊａｖａ　Ｂｅａｎ
１７０２　送信ＸＭＬ
１８０１　システム
１８０２　アプリケーション
１８０３　セッション
１８０４　リクエスト
１８０５　ハンドラ
１９０１　フロントコンポーネント
１９０２　ディスパッチャ
１９０３　アプリケーション
１９０４　セッション
１９０５　ハンドラ
１９０６　表示用ページ
１９０７　Ｄｉｓｐａｔｃｈ　Ｃｏｎｔｅｘｔ
１９０８　ＲｅｓｐｏｎｓｅＢｅａｎ
１９０９　コマンドマップ
１９１０　ページマップ
１９１１　ＲｅｑｕｅｓｔＢｅａｎ
１９１２　ＳｉｎｇｌｅＴｈｒｅａｄＭｏｄｅｌ
２００１　システム
２００２　アプリケーション
２００３　セッション
２００４　ハンドラ
２００５　ログイン処理メソッド
２００６　ログアウト処理メソッド
２００７　ユーザ変更メソッド
２００８　プロファイル編集メソッド
２００９　ユーザ登録メソッド

54

３００１　モデルオブジェクト
３００１ａ　表示用モデルオブジェクト
３００１ｂ　入力用モデルオブジェクト
３００２　モデルインタフェース
３００３　フレームワークエンジン
３００４　表示用ＪＳＰ
３００５　ブラウザ
３００６　フロントコンポーネント
３０１０　サーバ
３０１１　モデルフレームワーク処理部
３０１２　Ｗｅｂサーバ部
３０１３　バックエンド
３０１４　データベース
３０２０ａ、３０２０ｂ、３０２０ｃ　クライアント
４００１、４１０１　コンポーネントオブジェクト
４００２、４１０２　コンテナオブジェクト
４００３、４１０３　アクションオブジェクト
４００４、４１０４　スクリプト呼び出し部
４０１１　ValidInputTag オブジェクト
４０１２　ValidFormTagオブジェクト
４０１３ａ　CustomActionTag オブジェクト
４０１３ｂ　MyActionTag オブジェクト
５００１　ＣＰＵ
５００２　記憶部
５００３　入力部
５００４　出力部
５００５　Ｉ／Ｆ部
５００６　バス
５３０１　情報処理装置（コンピュータ）
５３０２　可搬型記録媒体
５３０３　ネットワーク
５３０４　プログラム提供サーバ
５４０１　アプリケーションサーバ
５４０２　データベースサーバ
５４０３　クライアント
５５０１　ＨＴＭＬ
５５０２　画面データ
５５０３　ロジック
５６０１　ＨＴＭＬ
５６０２　画面データ
５６０３　データ

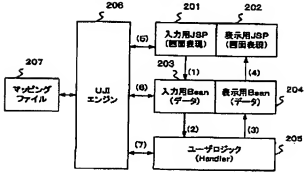
【図１】

本発明の概略を示す図



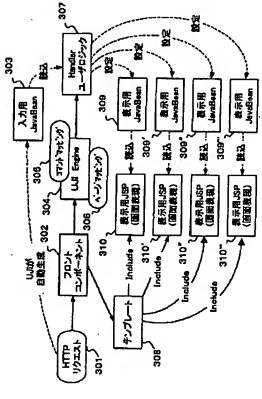
【図２】

本発明の原理構成を示す図



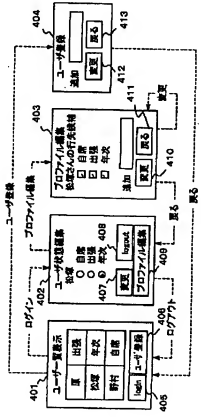
【図３】

本発明の第1の実施例のシステム構成を示す図



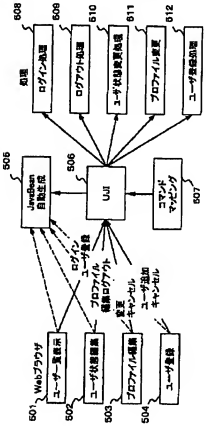
【図４】

本発明の第1の実施例のシステムの
具体的な動作の様子を示す図



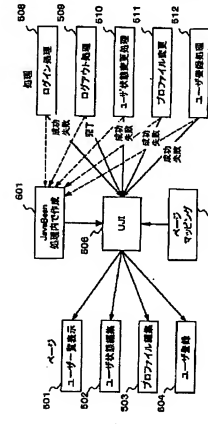
【図５】

図4に示すアプリケーションのシステム構成上その動作（クライアントからのリクエストを受信し、処理するまで）概要を説明する図



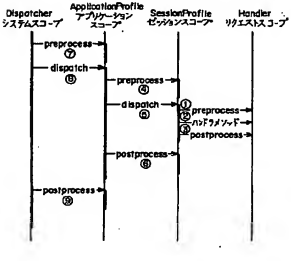
【図６】

図4に示すアプリケーションのシステム構成上その動作（クライアントからのリクエストを受信し、処理するまで）概要を説明する図



【図２１】

クライアントのリクエストを処理する場合のシーケンス図



【図７】

(a)，(b)はコマンドマッピングの一部、
(c)はページマッピングの一部を示す図

ユーザ状態編集画面からの分岐

入力データ	コマンド	処理
ユーザ状態	プロファイル編集	プロファイル変更処理
	ログアウト	ログアウト処理

(a)

入力データ	コマンド	処理
ユーザ状態	変更	ユーザ状態変更処理
プロファイル編集		プロファイル変更

(b)

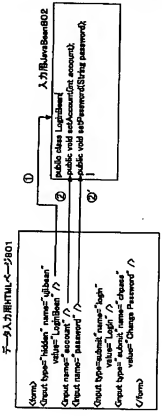
ログイン処理からの分岐

出力データ	処理結果	画面
ユーザ状態	ログイン成功	ユーザ状態編集画面
	ログイン失敗	ログイン失敗画面

(c)

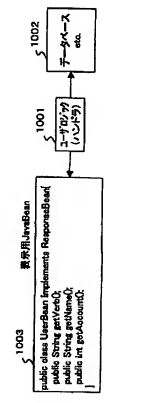
【図８】

クライアントからのリクエストデータを
入力用Java Beanに変換するプログラム記述を示す図



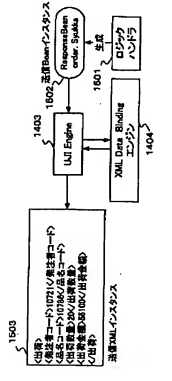
【図１０】

ユーザロジックが表示用データオブジェクト
を設定するプログラム記述を示す図



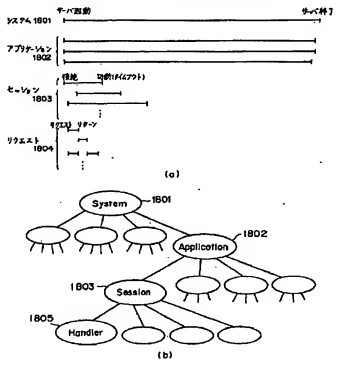
【図１５】

本発明の第2の実施例の
システム構成を示す図(その2)



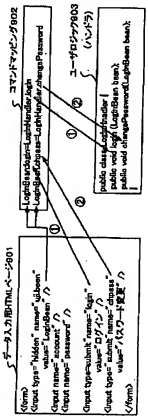
【図１８】

(a)は、本発明のシステムを構成するオブジェクトのスコープを説明する
図であり、(b)はオブジェクト間の相関関係を示す図



【図９】

コマンドマッピングを用いて実行すべき
ロジックの処理を決定するプログラム技術を示す図



【図２５】

テーブルモデルインタフェースの宣言例を示す図

```
public interface Table Model {
  public int getColumnCount();
  public int getRowCount();
  public Object getValueAt(int row ,int col);
  public String getColumnClass(int row ,int col);
  public String getRowClass(int row);
  public void setValueAt(Object value ,int row ,int col);
}
```

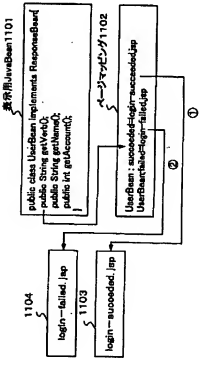
【図３１】

<name>のレンダラエレメントタグを使用したレンダラの定義例を示す図

```
<uomfTableRenderer type="column" clas="editable">
  <td><Input name="uomfname/>"
    value="uomfvalue/>" /></td>
</uomfTableRenderer>
```

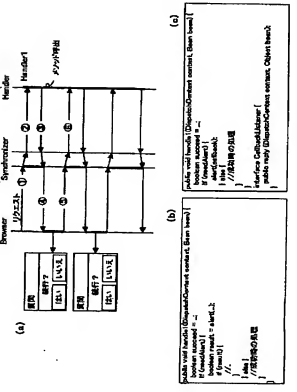
【図１１】

ユーザロジックが表示用データオブジェクト
を設定するプログラム記述を示す図



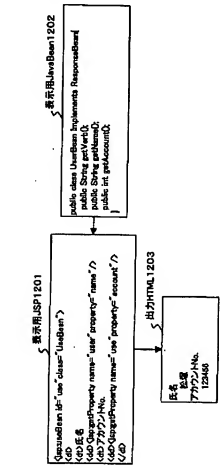
【図２３】

(a)は、本発明のアプリケーションサーバにおいてロジックハンドラから
ブラウザにリクエストを行う仕組みを説明する図であり、
(b)、(c)はそのプログラム記述例を示す図



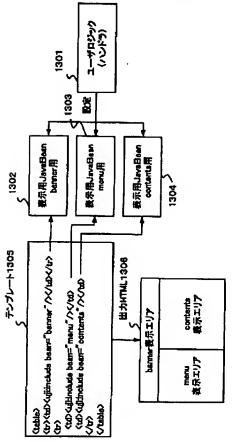
【図１２】

JSPから表示画面を出力するプログラム記述を示す図



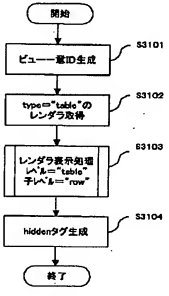
【図１３】

表示画面に複数の表示部品を配置する場合のテンプレートについて説明する図



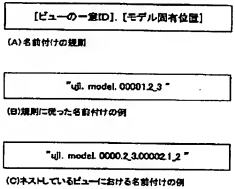
【図２７】

表示時の制御処理の処理内容を示すフローチャート（その１）



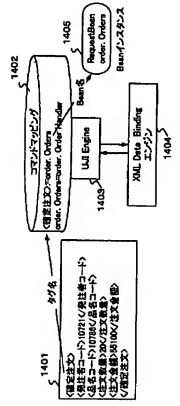
【図３０】

＜name＞のレンダラエレメントタグを説明する図



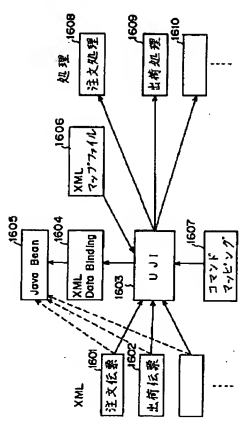
【図１４】

本発明の第2の実施例のシステム構成を示す図(その1)



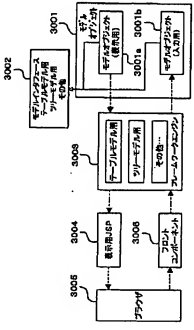
【図１６】

XMLファイルを処理するシステムの動作（XMLファイルを受信して、処理するまで）概要を示す図



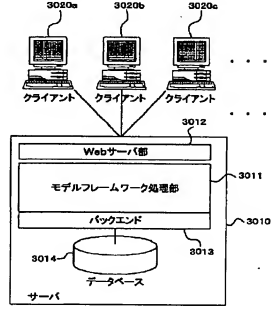
【図２４】

本発明の第3の実施例のシステム構成を示す図



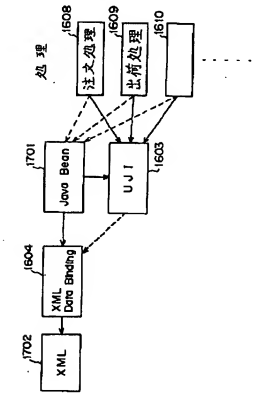
【図３５】

本発明の第3の実施例を実施するクライアントサーバシステムの例を示す図



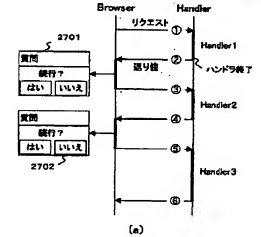
【図１７】

XMLファイルを処理するシステムの動作（XMLファイルを受信した時に、処理結果を返すまで）概要を示す図



【図２２】

(a)は、一般のアプリケーションサーバにおいてロジックハンドラからブラウザにコールバックする仕組みを説明する図であり、(b)はそのプログラム記述例を示す図

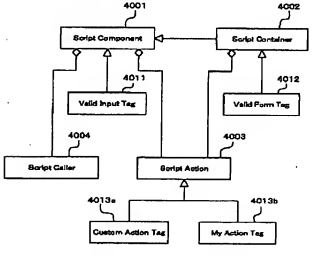



```
// 本来のイベントハンドラ
public void handle1(DispatchContext context, Bean bean) {
  boolean succeed = ...;
  if (needAlert) {
    //もとの画面用の設定
    // alert用Beanの設定
    return;
  }
  //成功時の処理
  // ...
}
//handle1でアラートを出した返り値のためのイベントハンドラ
public void handle2 (DispatchContext context, AlertBean bean) {
  boolean result = bean. getResult();
  if (result) {
    //...
  }
}
```

(b)

【図３７】

本発明の第4の実施例で使用されるオブジェクトの構成を示す図



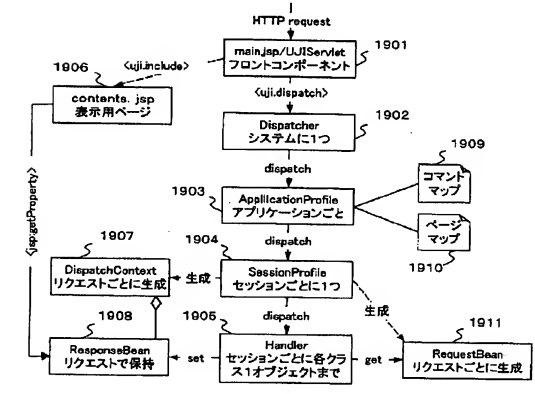
【図４２】

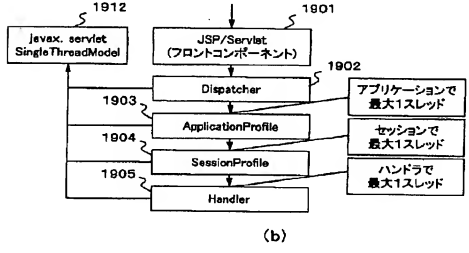
文字列の最小文字数をチェックする処理スクリプトの定義例を示す図

```
public class MyActionTag extends CustomActionTag{

protected int min =0;

public void setMin(Length0int min){
  this.min=min;
}

public void outputFunctionBody(Writer writer){
 if(min>0){
   writer.write("if(target.value.length<" +min+")\n");
   writer.write("  alert('" +min+ "文字以上入力してください');\n");
   writer.write("  return false;\n ");
   writer.write("}\n ");
 }
}
```

【図１９】

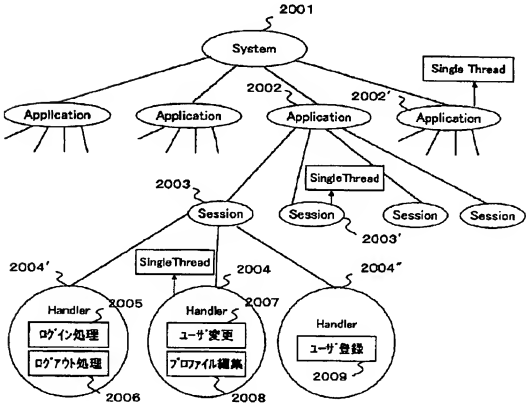
(ａ)は、クライアントからのリクエスト処理を説明する図
(ｂ)はSingleThreadModel の実装を説明する図





【図２０】

(ａ)は、SingleThreadModel の具体的な実装を示す図
(b)はSingleThreadModel の実装のプログラム記述例を示す図



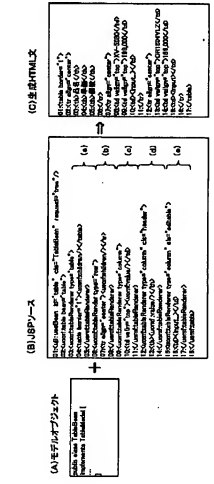
(a)

```
class   MySessionProfile extends SessionProfile implements SingleThread {
        ・・・・・・・・・・・・
        ・・・・・・・・・・・・
        ・・・・・・・・・・・・
}
```

(b)

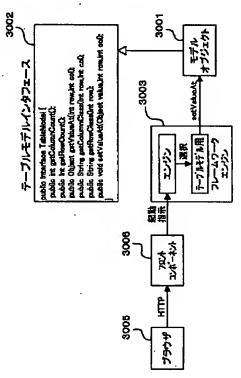
【図２６】

本発明の第3の実施例において、HTML文が生成される様子を説明する図



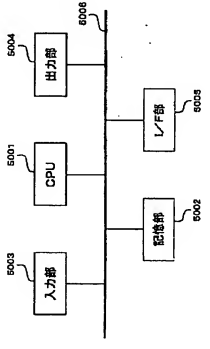
【図３３】

図24に示すシステムにおけるリクエスト時の動作の概要を説明する図



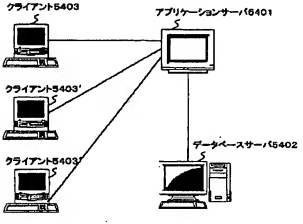
【図４５】

本発明の各実施例を実施するシステムに使用される、サーバの各処理エンジン及びクライアントの構成を示す図



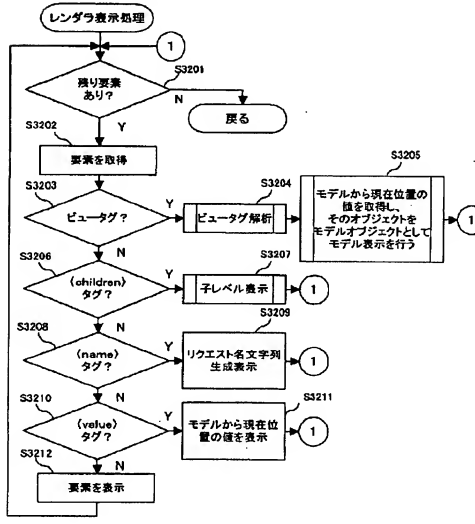
【図４７】

ビジネスアプリケーションの実施形態を示す図

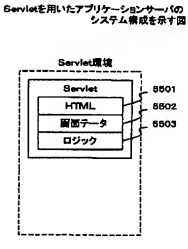


【図２８】

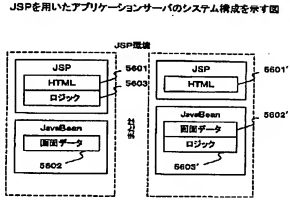
表示時の制御処理の処理内容を示すフローチャート(その2)



【図４８】

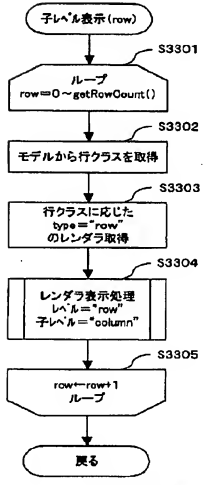


【図４９】



【図２９】

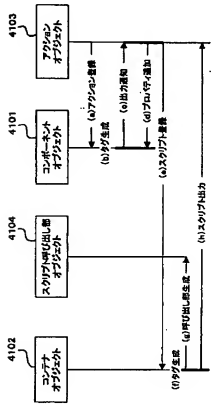
表示時の制御処理の処理内容を示すフローチャート（その3）



(A)子ﾚﾍﾞﾙがrowの時の表示処理　　(A)子ﾚﾍﾞﾙがcolumnの時の表示処理

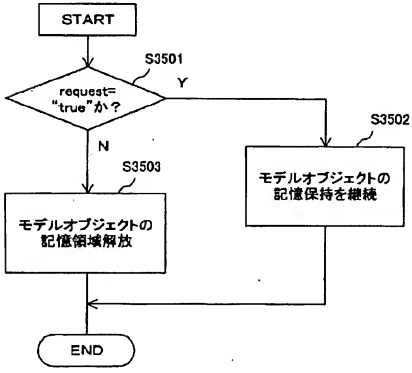
【図３８】

コンテンツ変換処理の概要を説明する図



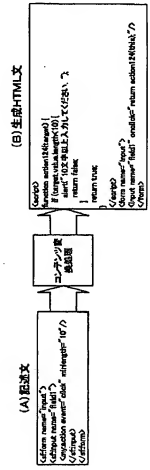
【図３２】

記憶領域管理処理の処理内容を示すフローチャート



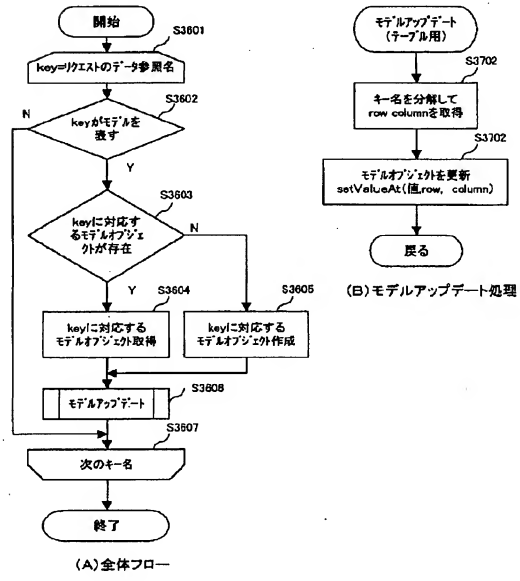
【図４１】

処理スクリプトを別に用意する場合の
スクリプト記述例を示す図



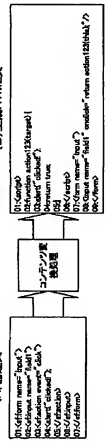
【図３４】

リクエスト時の制御処理の処理内容を示すフローチャート



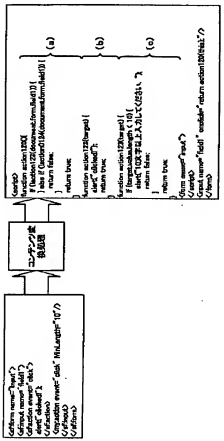
【図３６】

本発明の第4の実施例によってHTML文が生成される様子を示す図



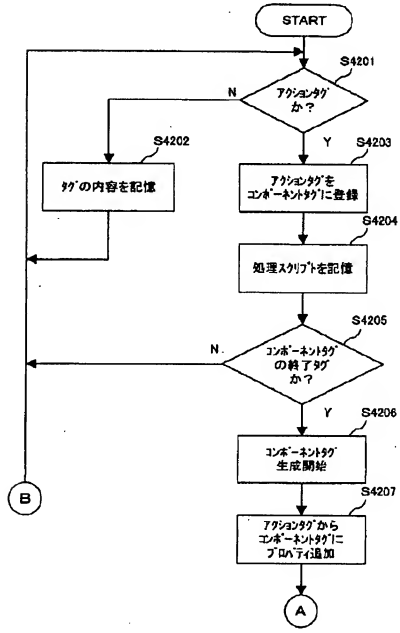
【図４３】

同一のイベントに対して複数のアクションが対応する場合のスクリプト記述例を示す図



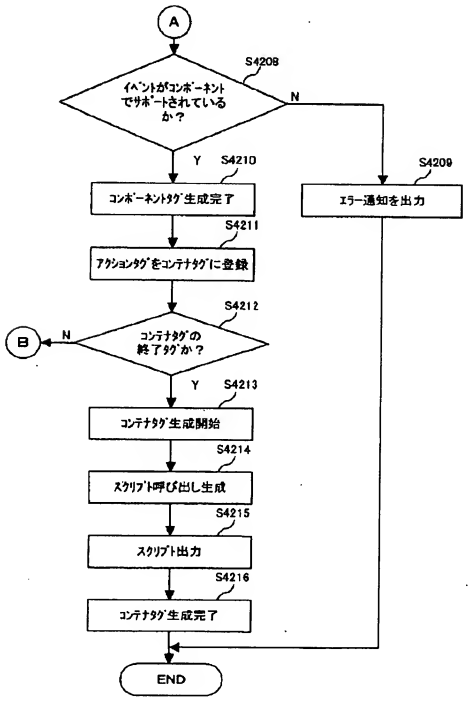
【図３９】

コンテンツ変換処理の処理内容を示すフローチャート（その1）



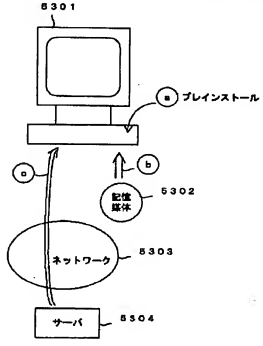
【図４０】

コンテンツ変換処理の処理内容を示すフローチャート（その2）



【図４６】

本発明に係わるソフトウェアプログラム等の提供方法を説明する図



【図４４】

コンテナタグでイベントが発生する場合のスクリプト記述例を示す図